

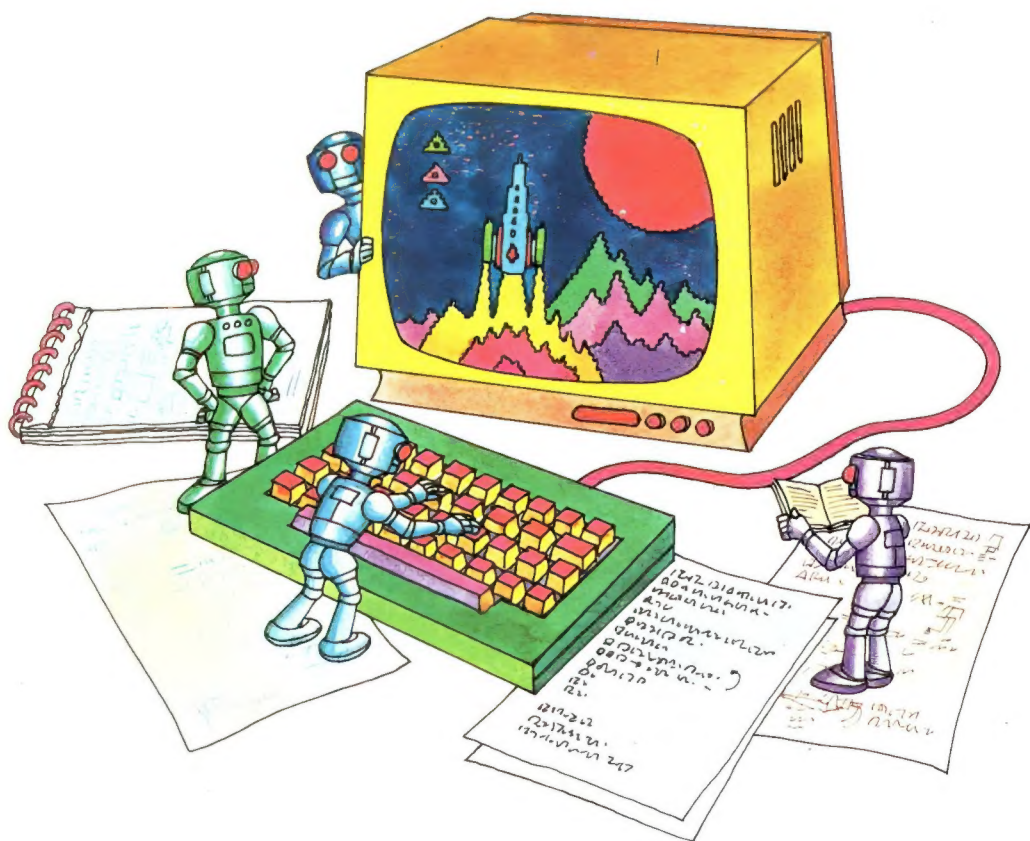
# ELSŐ KÖNYVEM A PROGRAMOZÁSRÓL



**Műszaki Könyvkiadó**  
**Novotrade Rt.**

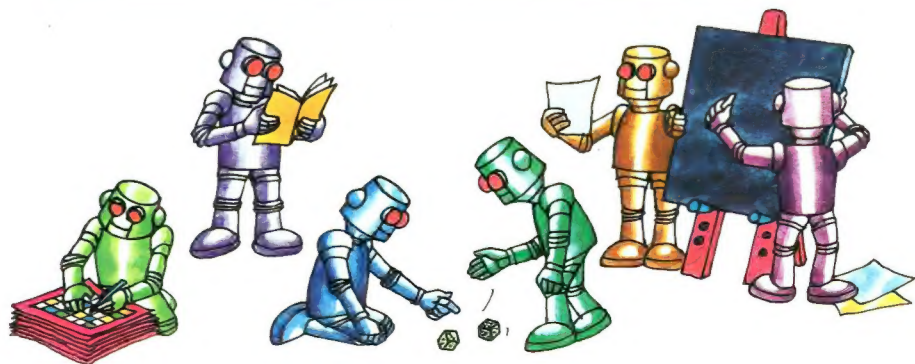
# PROGRAMOZÁS

## ELSŐ KÖNYVEM A PROGRAMOZÁSRÓL



# Tartalom

Könyvünkről röviden	2
Hogyan működik a számítógép?	4
Hogyan adunk utasítást a számítógépnek?	6
Hogyan írunk programokat?	8
Az első szavak BASIC-ben	10
Adatokat táplálunk a számítógépbe	12
Az INPUT használata	14
Mit tud még a PRINT?	16
Hogyan mérleget a számítógép?	18
Egy kicsit komolyabb programok	20
Rajzoljunk képeket!	22
Játsszunk véletlen játékokat!	24
Készítsünk ciklusokat!	26
Ciklusok trükkjei	28
Szubrutinok	30
Mi mindent lehet csinálni szavakkal?	32
Ábrák és szimbólumok	34
Mozgó képek	36
Egy furcsa költemény	38
Programozási ötletek	42
A fejtörők megoldásai	44
BASIC alapszavak	46
Hogyan tovább?	48
Tárgymutató	48



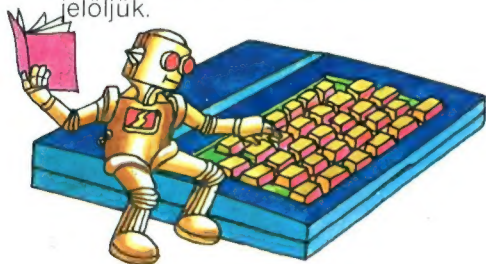


# Könyvünkről röviden

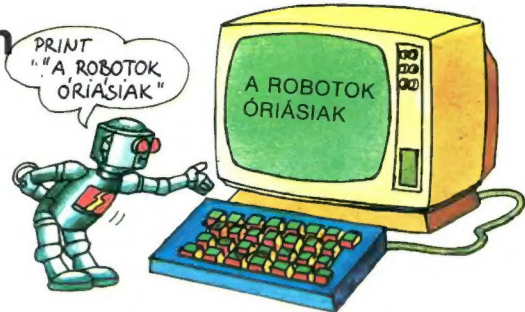
Ez a könyv a BASIC nyelvű programíráshoz nyújt segítséget a teljesen kezdők számára. A BASIC nyelvet használja a legtöbb számítógép; ezen a nyelven lehet utasításokat megfogalmazni úgy, hogy a gép is megértse.



A könyvet számítógép nélkül is megértheted, de természetesen sokkal jobb, ha a programokat ki tudod próbálni. A különböző számítógéptípusok némileg eltérő BASIC nyelvjárást ismernek. Ezért könyvünkben olyan kifejezéseket használunk, amelyek többségét a legtöbb számítógép megérti, azt a néhányat pedig, amelyik a szabványostól eltér, külön jelöljük.



Először általános tudnivalókat adunk a programozáshoz. A későbbiekben lépésenként fogsz megismerkedni a legfontosabb BASIC kifejezésekkel, amelyeket egy-egy rövid programmal mutatunk be.



Hogy némi gyakorlatra tegyél szert a programírásban, adunk néhány programfeladványt is, programírási javaslatokkal és hasznos módosítási ötletekkel. A programfeladványok megoldásait a 44 – 45. oldalon találod meg.

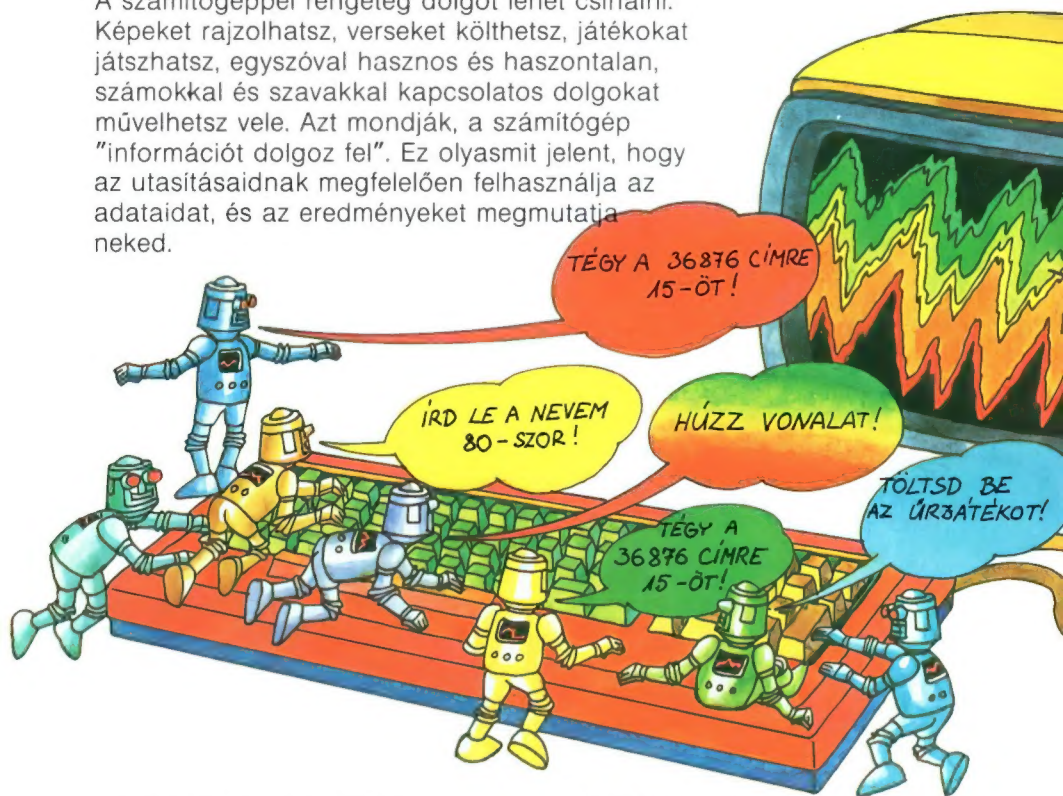
Könyvünk végén összefoglaljuk az előforduló számítógépes kifejezéseket és BASIC szavakat. Itt találsz majd néhány jó tanácsot a programíráshoz, és egy kis segítséget a hibakereséshez.



Ha van egy géped, feltétlenül próbáld ki programjainkat! Közben forgasd a gépkönyvet is, hogy jobban megismerd a géped működését! Az is lehetséges, hogy a te gépedre nem pontosan illenek az itt leírt szabályok. A legjobban úgy tanulsz meg a BASIC nyelvet, ha könyvekből és folyóiratokból minél több programot kimásolsz. A saját ízlésed szerint módosítva őket egyszerre csak azon kapod magad, hogy megírtad első önálló programodat.

# Hogyan működik a számítógép?

A számítógéppel rengeteg dolgot lehet csinálni. Képeket rajzolhatsz, verseket költhetsz, játékokat játszatsz, egyszóval hasznos és haszontalan, számokkal és szavakkal kapcsolatos dolgokat művelhetsz vele. Azt mondják, a számítógép "információt dolgoz fel". Ez olyasmit jelent, hogy az utasításaidnak megfelelően felhasználja az adataidat, és az eredményeket megmutatja neked.



A gépnek nagyon pontosan kell megadnod az utasításokat, hogy pontosan azt csinálja, amit szeretnél. Az utasítások egymásutánját programnak mondják, a gépnek adott információkat

pedig adatnak. A programot számítógépnyelven kell megírunk, pl. BASIC- ben, ezt megérti a gép, ha pontosan betartjuk a nyelvtani szabályait.

## Mikroszámítógépek

A legkisebb számítógépek a mikrók. A legtöbb mikró billentyűzettel ellátott, az írógépénél is kisebb készülék, amit a tv-hez lehet kapcsolni. Az utasításokat és az információkat a billentyűzetről gépeled be, a válaszok és eredmények pedig a tv-képernyőn jelennek meg. Némelyik mikrón beépített kis kijelző van, akár a zsebszámológépen. Másokhoz különleges képernyő tartozik, amit monitornak hívnak. A monitoron nem lehet fogni a tv-adást, de sokkal tisztább képet ad, mint a tv. A billentyűzet majdnem

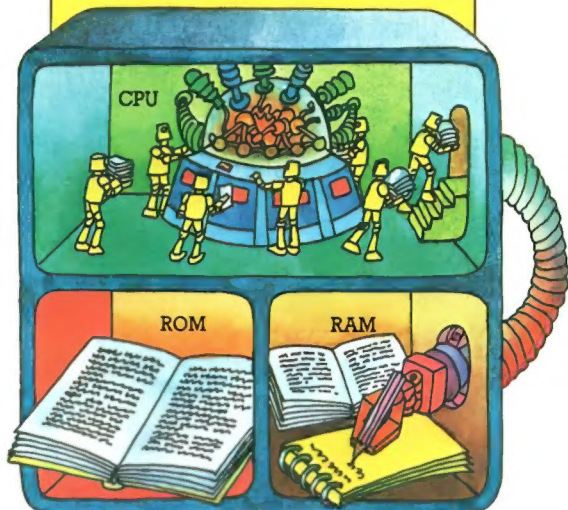
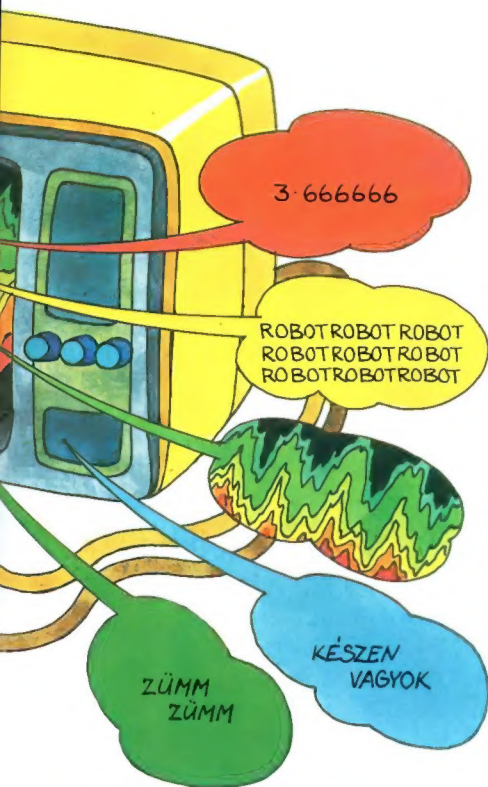


olyan, mint az írógépén, csak van még rajta egy-két különleges billentyű. Olyan mikró is van, amelyiken nem kell betűként begépelni az utasításokat, hanem minden billentyű egyben egy utasítást is jelent.

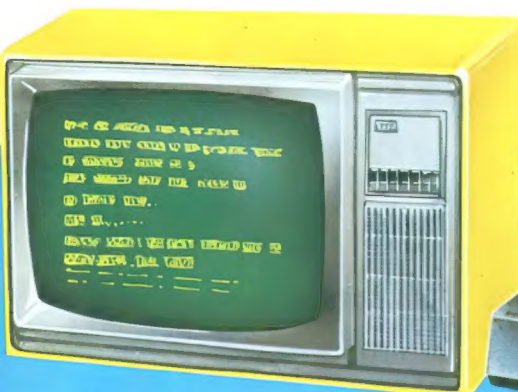


## Mi van a gép belsejében?

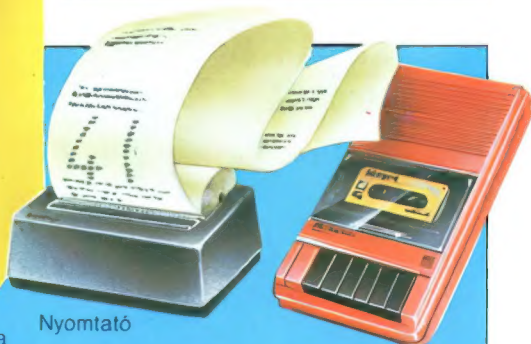
A mikró két fő részből áll, az egyik a központi feldolgozóegység (CPU), ez végzi a tényleges munkát, a másik pedig a memória, ahol a programok és az adatok vannak elraktározva.



Igazából a számítógépnek kétféle memóriája is van. Az egyik a csak olvasható memória, amiben a gépnek a műveletek elvégzéséhez szükséges saját programjai vannak, a másikat írásra és olvasásra is használhatja, ez való a programok és az adatok tárolására. Angol rövidítésük ROM és RAM. A ROM tartalma kikapcsolás után is megmarad.



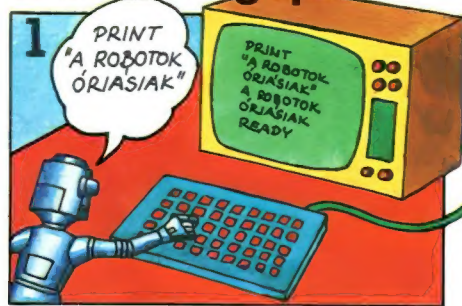
A mikrók legszokásosabb kijelzési módja a tv-képernyőn való megjelenítés. Ezenkívül sokszor látjuk használni a nyomtatót, amivel papírra is kiíratható a programjaidat vagy eredményeidet, és ez persze megmarad



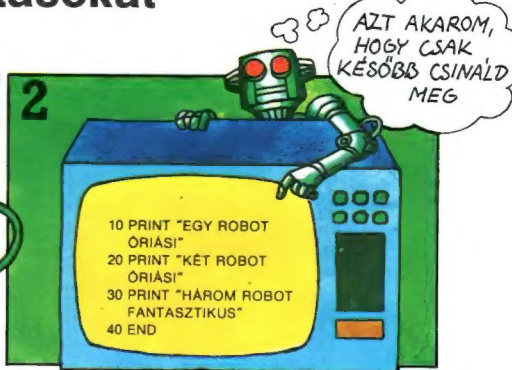
Nyomtató

a gép kikapcsolása után is. A másik a kazettás magnó, amire ki tudod menteni programjaidat és adataidat. Ha bármikor ismét szükség van rájuk, percek alatt vissza tudod tölteni a kazettáról a gépbe.

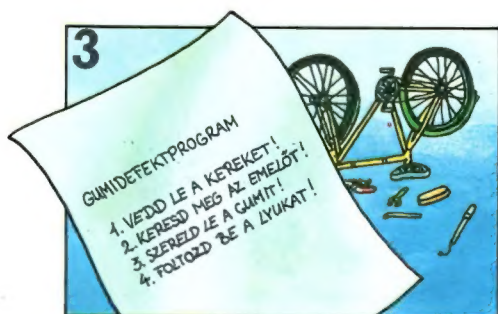
# Hogyan adunk utasításokat a számítógépnek?



Ha valamilyen feladatot el akarunk végeztetni a számítógéppel, ezt előbb el kell magyaráznunk neki. Ez történhet egy közvetlen paranccsal, amit rögtön



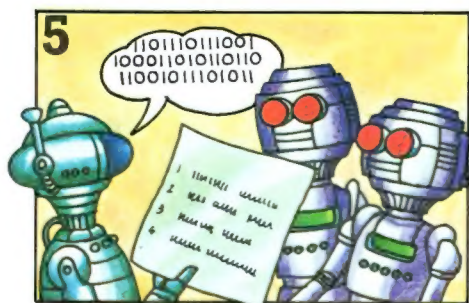
végrehajt, vagy egy utasításokból álló programmal, amelyet a memóriájában elraktároz, és addig nem vesz elő, amíg nem kap egy "láss neki" utasítást.



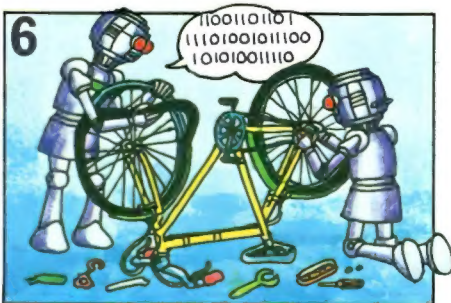
A program utasításait nagyon gondosan kell megadni! A számítógép ugyanis még a helytelen utasítások végrehajtását is megkísérli.



A gép nem érti az emberi nyelvet, csakis a sajátját. A szemközti lapon bemutatunk néhányat a számítógépnyelvek közül.



A számítógépen belül minden munkát a gép egy belső kód alapján végez el. A kód apró elektromos impulzusokból áll. Utasításaidat a gépen belül egy különleges értelmező program fordítja le számítógépes kódra.



A számítógépes kódban minden utasítást pontosan meghatározott elektromos impulzusok képviselnek. A számítógépes kódot úgy írhatjuk le, hogy 1-eset használunk, ha van impulzus, és 0-át, ha nincs.



# Számítógépnyelvek

Írhatnánk programokat a számítógép saját kódjában is, de ez nagyon nehéz lenne. Ehelyett különleges számítógépnyelveket használunk, amelyek közelebb állnak az emberi nyelvhez, ezért magas szintű nyelveknek hívják őket. Ezeket azután a számítógép le tudja fordítani a saját kódjára. Több száz magas szintű nyelv van. A BASIC az egyik legáltalánosabb nyelv. Betűi a "Beginner's All-purpose Symbolic Instruction Code" (kezdők általános célú szimbolikus utasításkódja) kifejezés kezdőbetűiből állnak össze, de nemcsak kezdők számára találták ki. Most pedig három, különböző nyelven írt példát mutatunk be:

## Névprogram

```
10 PRINT "HOGY HÍVNÁK?"
20 INPUT N$
30 IF N$ = "LAJOS" THEN
  PRINT "HELLO"
40 IF N$ = "GYURI" THEN
  PRINT "MEHETSZ"
50 END
```

## Pénzprogram

```
PROCEDURE PENZ;
VAR HONAP: INTEGER
BEGIN (*SZÁMOLJ*)
  FOR HONAP = 1 TO
    OSSZES HONAP DO
    ESEDEKES OSSZEG :=
    (ESEDEKES OSSZEG +
    VISSZAFIZETES) *
    (1 + HAVI KAMATLÁB)
  END; (*PENZ*)
```

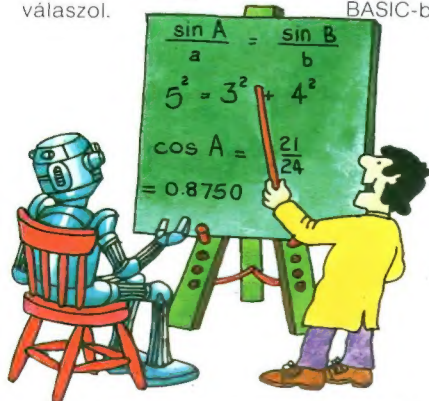
## Földrajzi program

```
T: MI FINNORSZÁG FŐVÁROSA?
*INPUT A$AS
T(LEN(A$) - 0) : NOS, MIRE GONDOLSZ?
JC: (H A
M: KOPE STOCK OSLO, GOTHEN LAP
TY: LANGYOS, DE MEG NEM MELEG
  - PRÓBÁLD MEG ÚJRA!
JY: (H A
M: HELSINKI HELSING
TY: IGEN -> HELSINKI.
```

Ez egy rövid BASIC nyelvű program. A 10-es sor arra utasítja a gépet, hogy írja ki a képernyőre a "HOGY HÍVNÁK" mondatot. Ezután a gép a memóriájában elraktározza a választ, és ha Lajos vagy Gyuri nevet írsz be, egy üzenettel válaszol.

Ez egy Pascal nyelven írt program. Pascal híres francia matematikus volt. A programrészlet valamilyen havi elszámolást mutat. Sokak szerint könnyebb jó és világos programokat írni Pascal-ban, mint BASIC-ben.

Ez itt PILOT nyelven íródott. Gyakran használják valamilyen tantárgy tanulását segítő oktatóprogramok írásához. Azért jó ilyesmire, mert felismeri a nem egészen pontos válaszokat is.



- 11. Bb3, He5
- 12. 0-0-0, Hc4
- 13. Bxc4, Vxc4
- 14. h5, Hxh5

Első pillantásra a számítógépnyelvek furcsának és nehéznek tűnnek, de azért akár a többi nyelv, ezek is megtanulhatók. Sok más esetben is szükség lehet különleges nyelvekre. Pl. a matematikában a gondolatok és a képletek leírására olyan

jelrendszerünk van, amit hétköznapi szavakkal csak hosszadalmasan lehetne elmagyarázni. Megint más jelekkel írjuk le a sakkfigurák mozgását, vagy a kottában a hangjegyeket.





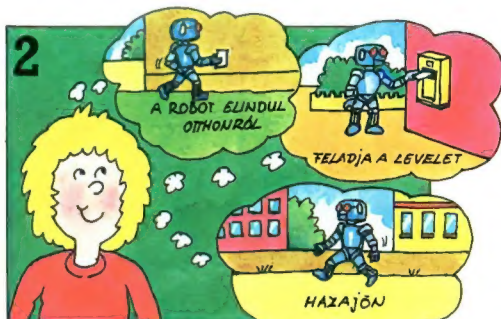
# Hogyan írunk programokat?

Egy program olyan, mint egy játékszabály, vagy mint egy süteményrecept. Ha hibásak a szabályok vagy hibás a recept, nem tudod lejátszani a játékot, ill. nem tudsz jó tésztát sütni. Épp így van a számítógépnél is: az eredmény attól függ, hogy milyen utasításokat adsz. Mielőtt programot írsz, nagyon alaposan meg kell fontolnod, mit is akarsz csinálni, és hogy a várt eredményre juss, meg kell határozni a főbb lépéseket.

## Levélprogram



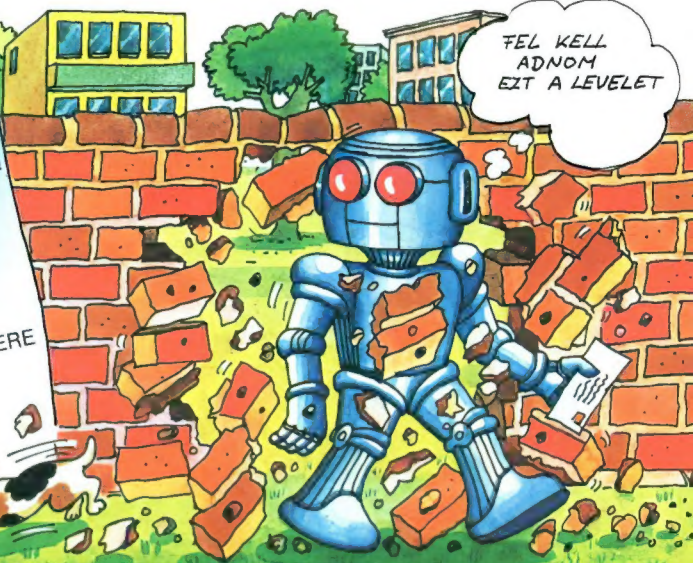
Képzeld el, hogy egy robotot akarsz a postára küldeni. Nem elég azonban egyszerűen megkérni, hogy "add fel a levelet", mert egy ilyen kérést nehéz volna felfognia.



Pontosan meg kell mondani neki, hogy mi a teendő. A számítógépnek is pontosan meg kell adni minden egyes lépést.

3

MENJ EL OTTHONRÓL!  
MENJ A BEJÁRATI AJTÓHOZ!  
NYISD KI AZ AJTÓT!  
MENJ KI ÉS CSUKD BE AZ  
AJTÓT!  
KERESD MEG A POSTALÁDÁT!  
ADD FEL A LEVELET!  
TEDD A LEVELET  
A NYILÁSHOZ!  
DORD BE A LÁDÁBA!  
GYERE HAZA!  
FORDULJ MEG!  
UGYANAZON AZ ÚTON GYERE  
HAZA!  
GYERE BE!  
CSUKD BE AZ AJTÓT!



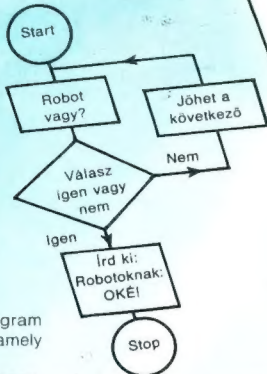
Programíráskor az utasításokat olyan egészen apró lépésekre kell bontani, amelyek már lefordíthatók a robot nyelvére. A robot engedelmesen követi utasításaidat, akkor is, ha azok hibásak

vagy hiányosak. A programhibákat angolul bugnak, poloskának hívják. (Ezért terpszkedik egy a szemközti rajzon.) Tényleg, van is mit "bogarászni" a programhibákon!

# Folyamatábrák

Amikor programot írsz, segítségedre lehet egy olyan vázlat is, amiben logikusan összegyűjtöd a fő lépéseket. Az ilyen vázlatokat folyamatábrának nevezzük. A folyamatábrák a végrehajtandó lépések sorrendjét is jelzik.

Robotteszt

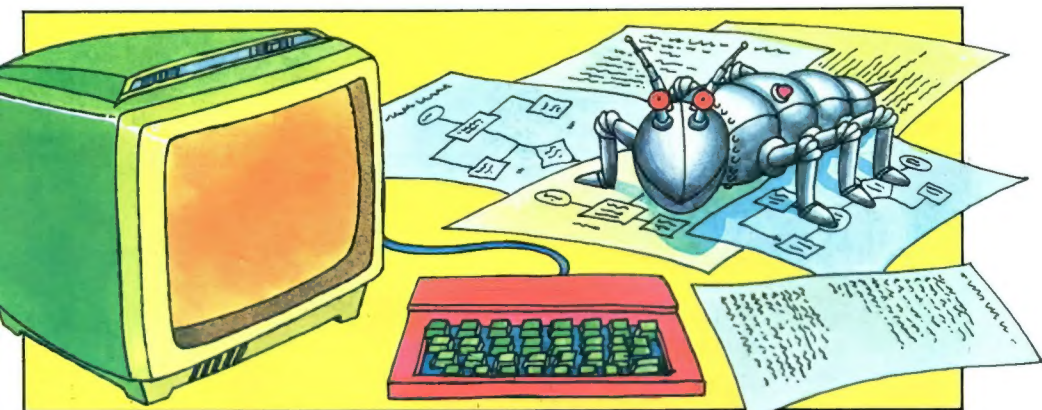


Ez egy olyan program folyamatábrája, amely megtanítja a számítógépet, hogy csak robotnak írjon ki OKÉ üzenetet.

A folyamatábrán a program különböző lépéseit különböző alakú "dobozok" szemléltetik. A program eleje és a vége egy-egy kerek doboz, az utasításokat pedig, amelyek megmondják a gépnek,



hogy mit csináljon, téglalap alakú dobozokba írjuk. A döntéseknél, ahol a gép a választól függően mást és mást tesz, rombuszokat rajzolunk. A vonalak a számítógép lehetséges útvonalait mutatják.



Miután a program minden elvi részletét kidolgoztuk, következik a BASIC-re való átfogalmazás és a kipróbálás. A program valószínűleg nem fog azonnal működni, biztosan lesz még benne hiba. Lehet, hogy egyszerű gépelési hiba, de az is lehet, hogy komolyabb, logikai hiba. Csak

akkor vagy kész egy programmal, ha megtaláltad az összes hibát, és ki is javítottad őket.\* Néha egy-egy programhiba nem várt eredményt hoz, ami esetleg jobban tetszhet, mint az eredeti elképzelésed.

\* A 42 – 43. oldalakon találsz egy-két ötletet a hibák kikeréséséhez!



# Az első szavak BASIC-ben

A BASIC legtöbb szava közönséges angol szó, így annak, aki tud angolul, nem nehéz megértenie. A **PRINT** (NYOMTASS) pl. azt jelenti, hogy "írd ki a képernyőre", a **RUN** (FUSS) azt, hogy "hajtsd végre a programot", az **INPUT** (BEVITEL) pedig azt, hogy "olvass be egy választ". A következő két oldalon a **PRINT** használatát tanulhatod meg. A legtöbb személyi számítógépnek beépített BASIC nyelvű "tolmácsa", azaz értelmezője van, így már bekapcsoláskor kész arra, hogy BASIC-ben programozzunk.



Ha bekapcsolod a mikrót, a legtöbbször rögtön megjelenik a képernyőn egy-két szó és egy kis világító jel, más szóval kurzor. A kurzor azon a ponton villog, ahová a következő begépelte betű fog kerülni.



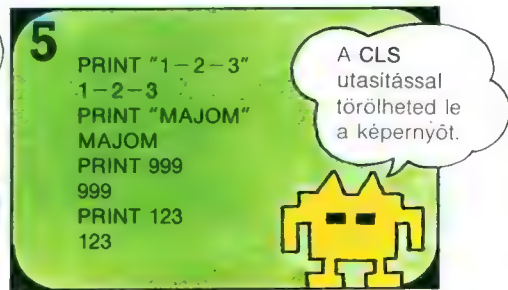
A számítógépet úgy kérheted meg arra, hogy írjon ki valamit a képernyőre, hogy egy **PRINT** utasítást adsz, és utána idézőjelek közé írod a szöveget. Pl. a **PRINT "CSIGA"** hatására megjelenik a CSIGA kiírás.



A számítógép addig nem hajtja végre utasításaidat, amíg nem nyomod meg a megfelelő **NEWLINE** (ÚJ SOR) vagy **RETURN** (VISSZATÉRÉS), vagy **ENTER** (BEADÁS) billentyűt.



A számítógép bármit kiír a képernyőre, amit idézőjelek között begépelsz. Ezek lehetnek betűk, számok, szavak vagy különleges jelek. Figyeld meg, hogy az idézőjeleket nem írja ki!



A számok kiírásához nincs szükség idézőjelekre. Ha le akarod törölni a képernyőt, a legtöbb mikrón a **CLS**-t kell begépelni. (Clear Screen: képernyőtörölés.) Próbáld ki, és ha nem működik, nézd meg a gépkönyvedet!

# Egy BASIC program

Eddig csak közvetlen parancsokat adtunk, most lássunk hozzá a programíráshoz. A programokban minden utasításor egy számmal kezdődik. Ennek hatására a gép elraktározza az utasításokat a memóriájában, és addig nem hajtja végre azokat, amíg meg nem kap egy "láss neki" parancsot. A szemközti lap szám nélküli utasításait rögtön végrehajtotta a gép. Nézzünk most egy rövid programot, ami betűből és írásjelekből egy arcot rajzol a képernyőre!

A legtöbb számítógépen a 0 át van húzva.

```
10 PRINT "/////"
20 PRINT "I      I"
30 PRINT "I( . . )I"
40 PRINT "I -L I"
50 PRINT "VVVV"
60 END
```

A sorszámok általában tízesével nőnek, így még pótlólag is beszűrhet az utasításokat, és nem kell átszámolni a programot.



Sok gépnél nincs szükség erre a sorra.

A programíráskor minden sor után megnyomod a **NEWLINE**-t. A sorok megjelennek a képernyőn, de a gép nem hajtja végre az utasításokat, amíg nem írod be, hogy **RUN**. Légy óvatos az

O betűvel és a 0-val, nehogy összecseréld őket! Minden gépen van egy javítóbillentyű – **RUBOUT (RADÍROZNI)** vagy **DELETE (TÖRÖLNI)** –, amivel a programsoron belül törölhetsz.

```
RUN
//////
I      I
I( . . )I
I -L I
VVVV
```

A számítógép pontosan kiírja, amit idézőjel között gépetl be, még a betűközöket is.

Amikor minden sort begépettél, és gondosan ellenőrizted, hogy nincs hiba a gépelésben, felszólíthatod a gépet, hajtassa végre a programot: **RUN**, majd **NEWLINE**.

```
RUN
MISSING" (HIÁNYZÓ)"
LIST
```

Hibaüzenet

A javításhoz visszakaphatod a programlistát a **LIST** és a **NEWLINE** utasítással.

Ha a program nem működik, vagy a kép nem olyan, amilyet szeretnél, újra a képernyőre hívhatod a programot. Ehhez a **LIST (LISTÁZZ)** szót kell begépelned. Némelyik számítógép pontosan megadja, hogy melyik sorban mi a hiba.

## Javítóprogramok

```
RUN
MISSING"
LIST
10 PRINT "/////"
20 PRINT "I      I"
30 PRINT "I( . . )I"
40 PRINT "I -L I"
50 PRINT "VVVV"
60 END
```

Nincs idézőjel

A számítógép a legtöbb hibáról hibaüzenetet ad. A hibaüzeneteket a gépkönyv részletezi. A hibajavítás legbiztosabb módja, ha az egész sort újra begépeljük. Az új sor felülírja a régit. Egy egész sort egyszerűen úgy tudsz törölni,

```
RUN
MISSING"
LIST
10 PRINT "/////"
20 PRINT "I      I"
30 PRINT "I( . . )I"
40 PRINT "I -L I"
50 PRINT "VVVV"
60 END
50 PRINT "VVVV"
```

Ugyanaz a sor de most már helyesen.

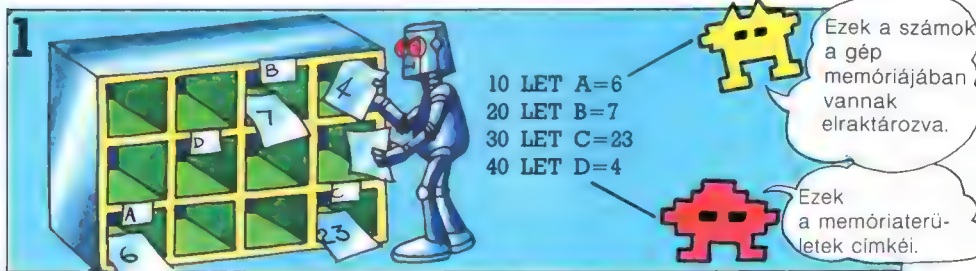
hogy beírod a sorszámát és a **NEWLINE**-t. Mindegyik gépnek megvan a saját szava a javításhoz, vagy a sor egyes részeinek a megváltoztatására, ilyen pl. az **EDIT** vagy a **COPY**. A gépkönyvedből nézz utána a részleteknek.





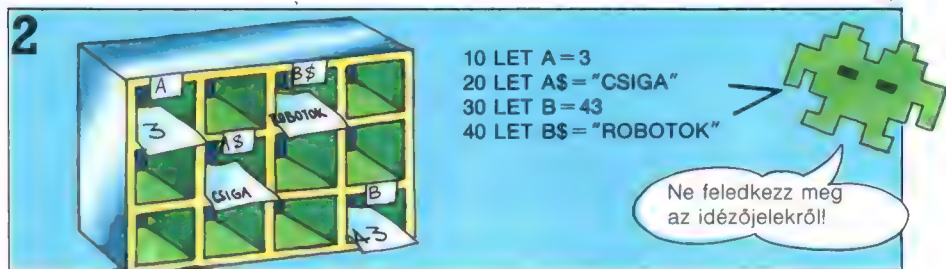
# Adatokat táplálunk a számítógépbe

Ha valami hasznosabbat vársz a géptől, és nem elégszel meg azzal, hogy ezt-azt kiíratsz a képernyőre, akkor olyan adatokkal kell ellátnod, amelyeket fel tud dolgozni. A számítógép az adatokat szépen elraktározza, és bármikor a rendelkezésedre bocsátja, ha kéred.



Ha egy adatot betesz a gép memóriájába, meg kell jelölni, meg kell címkézni, hogy később megtalálható legyen. Címkenek az ábécé betűit használhatjuk. Egy-egy memóriaterület

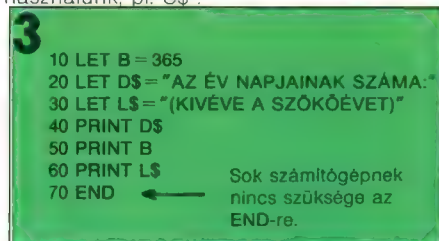
címkezésére és a számok elhelyezésére a **LET (LEGYEN)** szócskát használjuk. A címkézett memóriaterületet változónak hívjuk, mert különböző időpontokban változó adatokat hordoz a programban.



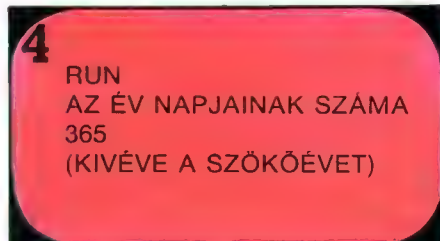
A betűk, számok és írásjelek (összefoglaló néven karakterek) elraktározásához másfajta címkeket használunk.

A karakterek láncokká fűzhetők össze, ahogy pl. a betűk szavakat alkotnak. A címkézésükre dollárjellel ellátott betűket használunk, pl. C\$\*.

A szövegváltozókat vagyis karakterláncokat ugyanúgy, mint a számváltozókat, a **LET** begépelésével raktározzuk el egy memóriaterületen, de mindig idézőjelek közé kell tenni, ahogy az a fenti képen is látható.



A változó kiírására a változó neve előtt kell begépelned a **PRINT** szót, pl. **PRINT A\$**. Ez a rövid program a B, a D\$ és az L\$ változókkal működik.



A programot annyi-szor futtathatod le, ahány-szor csak akarod. A számítógép minden alkalommal ugyanazt fogja kiírni. A változó-kban az adatok mindaddig ugyanazok maradnak, amíg nem változtatod meg őket.

\* Úgy mondjuk ki, hogy "C dollár".

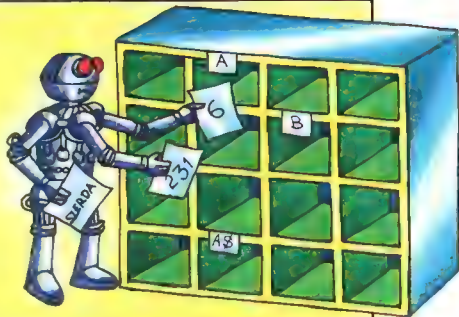
## Egy másik módszer

10 READ A  
20 READ B  
30 READ A\$  
40 DATA 6, 231, SZERDA

A számokhoz és a betűkhöz a megfelelő címkefajta! használj!

Vesszők

Néhány számítógépnél a DATA-n belüli szavakat idézőjelbe kell tenni.



A READ (OLVASD) és a DATA\* (ADAT) kulcsszavakat is használhatjuk információk tárolására. A READ sorok megcímkézik a memóriarekeszeket, a DATA sor pedig adatokat tárol.

Ha lefuttatod a programot, a számítógép minden egyes adatot rendben elraktároz egy memóriaterületen. Az adatok közé vesszőt kell tenni, hogy a gép tudja, melyik milyen hosszú.

## Néhány program

1 10 READ Q  
20 READ X\$  
30 DATA 24, SAJTKUKAC  
40 PRINT Q  
50 PRINT X\$  
60 END  
RUN  
24  
SAJTKUKAC

Vessző

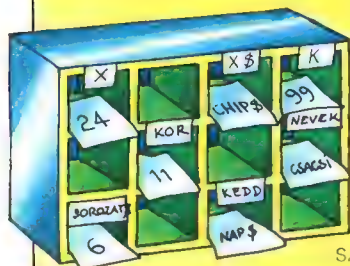
Ez egy adatot en

Itt van két program, az egyik a READ-et és a DATA-t, a másik pedig a LET-et használja a memóriában való raktározásra

2 10 LET A\$ = "A ROBOTOK ÓRIÁSIÁK"  
20 LET B\$ = "(AKI BÍRJA)"  
30 LET C\$ = "A FÉMKOLOSSZUSOKAT"  
40 PRINT A\$  
50 PRINT B\$  
60 PRINT C\$  
70 END  
RUN  
A ROBOTOK ÓRIÁSIÁK  
(AKI BÍRJA)  
A FÉMKOLOSSZUSOKAT

Idézőjelek!

## Még egy kicsit a változókról



Számváltozó

Szövegváltozó

Ezeket nem használhatod változócímkéknek, mert BASIC szavak vannak elrejtve bennük!



A változók címkézett területek a gép memóriájában, itt tárolja az adatokat. Azt a változót, amelyik csak számokból áll, számváltozónak, azt pedig, amelyik betűkből és szimbólumokból áll, szövegváltozónak nevezzük. A változók

tartalma a program során változhat. Néhány számítógépen szavakat is használhatsz változócímkéknek, de a BASIC szavait nem, mert ez összezavarná a számítógépet.

\* Ez a módszer nem alkalmazható a ZX 81 számítógépnél!



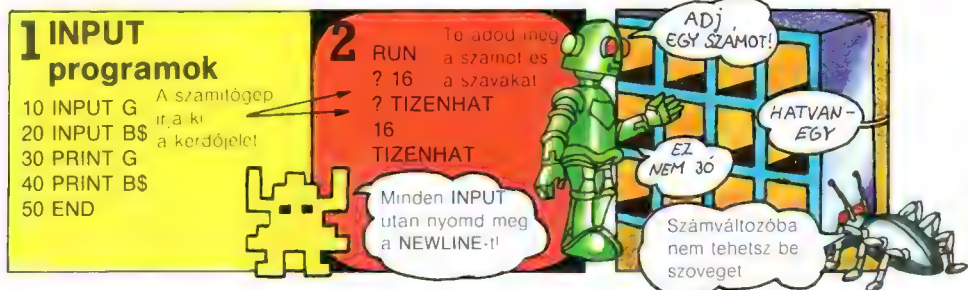
# Az INPUT használata

Az INPUT (BEVITEL) egy másik módszer, amivel adatot tárolhatunk a számítógépben. Segítségével a program futása közben, minden futtatáskor mást és mást táplálhatunk be a gépbe.



Az INPUT után egy címke következik, pl. A számoknál, és AS\$ a szövegeknél. Amikor a gép futtatás közben az INPUT szót találkozik, megcímkezi egy memóriaterületet, és kéri az adatot.

Illyenkor általában egy kérdőjelet ír ki a képernyőre. Ezután be kell gépelned az adatot, amit a gép elraktároz a memóriában, majd folytatja tovább a programot.



A 2. ábra azt mutatja, mi történik, ha lefuttatjuk az 1. ábra programját. Amikor a gép a 10. sorban találkozik az INPUT szóval, kiír egy kérdőjelet a képernyőre, és vár, míg begépsz egy számot G értékének.

Ezután egy újabb kérdőjelet ír ki a 20. sor INPUT utasítása nyomán. Ekkor szavakat vagy szimbólumokat kell begépelned, mert a BS címke azt jelzi a gépnek, hogy szövegre kell várnia.



Ha van géped, ird be ezt a programot, majd a RUN-t, hogy elinduljon! A számítógép kérésére ird be a nevedet és a korodat (lehet kitalált nevet és valótlán számot is, ahogy a példánk mutatja).

Próbáld ki többször különböző adatokkal, mindig beírva a RUN-t, hogy újra elinduljon a program! Látni fogod, hogy a gép mindig pontosan azt írja ki, amit az NS-ba és az A-ba tettél.

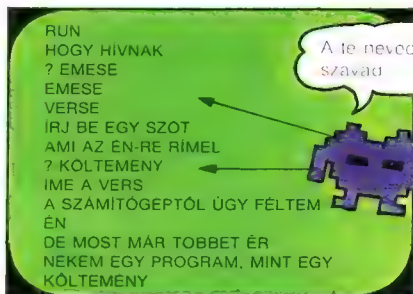
# Versíró program

Most már tudsz annyit a BASIC-ből, hogy egy kis verset is írhatasz a számítógéppel. Következik egy versíróprogram, amely a **PRINT** és az **INPUT** használatára épül

```
10 PRINT "HOGY HÍVNAK"  
20 INPUT N$  
30 PRINT N$ ]  
40 PRINT "VERSE"  
50 PRINT "ÍRJ BE EGY SZÓT"  
60 PRINT "AMI AZ ÉN-RE RÍMEL"  
70 INPUT A$  
80 PRINT "ÍME A VERS"  
90 PRINT "A SZÁMÍTÓGÉPTŐL ÚGY  
FÉLTEM ÉN"  
100 PRINT "DE MOST MÁR TÖBBET ÉR  
NEKEM EGY PROGRAM, MINT EGY"  
110 PRINT A$ ]  
120 END
```

Ez a sor  
kiírja a nevedet

Ez a sor  
kiírja a szavadat.



Gépeld be a **RUN**-t és próbáld meg újra egy másik szóval!

A program alapján a számítógép megkérdi a nevedet, azután a válaszodat beleteszi az **N\$**-ba, és a 30 sorban kiírja. Ezután elraktározza azt a szót, amit az

**A\$**-ba tettél, és a vers részeként írja ki a 110-es sorban. Ezt a programot is próbáld ki egy párszor úgy, hogy a 70-es sornál mindig más szavakat táplálsz be!

## Programfejtő

Tudnál olyan programot írni, amelyben a számítógép megkérdezi a nevedet, utána azt írja ki, hogy szia, majd a nevedet, és végül valami üzenetet a számodra?

## Teendők a program beírása során

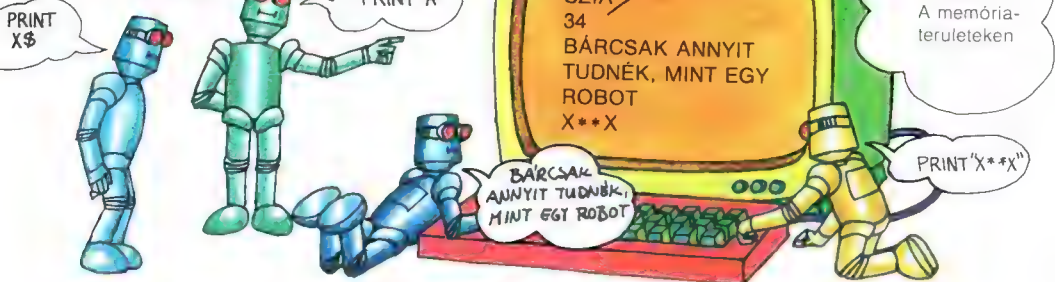
1. Mielőtt begépelnel egy programot, írd be, hogy **NEW (ÚJ)**. Ez minden régi programot és változót kitorol a gép memóriájából!
2. A program beírásakor ne felejsd el minden sort **NEWLINE**-nal lezárni!
3. Amikor beírtad a programot, ellenőrizd valamennyi sort a képernyőn, hogy kiszűrd az esetleges gépelési hibákat! Győződj meg arról is, hogy egyik sor sem hiányzik!
4. Ezután begépelheted a **CLS**-t (vagy a számítógéped hasonló szavát), ami letörli a programot a képernyőről. Most már el lehet indítani a programot: gépeld be a **RUN**-t!
5. Ha újra szeretnéd látni a programlistát, gépeld be a **LIST**-et! Ha egy meghatározott sort akarsz kiírtni, általában elég a **LIST** és a sorszám begépelése, de ezt ellenőrizd a gépkönyvedben, mert ez az utasítás a különböző számítógépeknél eléggé változó.
6. Ha futás közben meg akarod állítani a programot, gépeld be a **BREAK (MEGSZAKÍTÁS)** vagy az **ESCAPE (KILÉPÉS)** szót, de ezt mindenképpen meg kell nézned a gépkönyvben. Néhány gépnél az **ESCAPE** az egész programot kitorli. Ha újra akarod indítani a programot, **RUN**-t kell beírnod

A 42 – 43. oldalon találsz néhány hibakeresési tanácsot!





# Mit tud még a PRINT?



Eddig a **PRINT**-et csak szavak és számok megjelenítésére és változók tartalmának kinyomtatásához használtuk. Most megtanulhatod, hogyan kell vesszőkkel és pontosvesszőkkel a képernyőn elválasztani egymástól a dolgokat.

A **PRINT** segítségével számításokat is végeztethetsz a számítógéppel. A lap alján majd meglátod, hogyan. A szemközti lapról azt is megtudod, hogy még mi mindenre jók a változók.

## Vesszők és pontosvesszők

10 PRINT "MEG VAGYOK"  
20 PRINT "RITKÍTVÁ" ← Vessző

10 PRINT "ÖSSZE VAGYOK"  
20 PRINT "ZSÚFOLVA" ← Pontosvessző

10 PRINT "NAGYON MEG VAGYOK"  
20 PRINT  
30 PRINT "RITKÍTVÁ"

MEG VAGYOK RITKÍTVÁ

ÖSSZE VAGYOKZSÚFOLVA

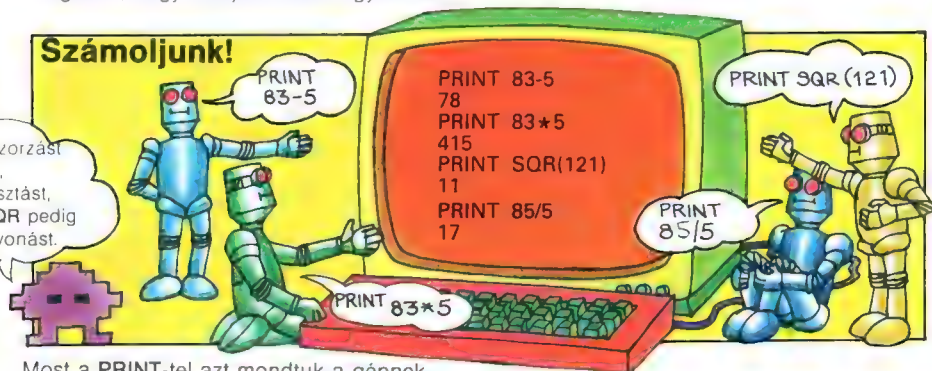
NAGYON MEG VAGYOK  
RITKÍTVÁ  
Egy önmagában álló PRINT ilyen üres sort eredményez.

Ezek a sorok bemutatják, hogyan lehet vesszőkkel és pontosvesszőkkel megmondani a gépnek, hova írja ki a következő betűt. A vessző azt mondja meg neki, hogy menjen arrébb egy kicsit

a képernyőn, a pontosvessző azt, hogy maradjon ott, ahol van. Az ábra azt mutatja, hogyan jelennek meg ezek a sorok a képernyőn.

## Számoljunk!

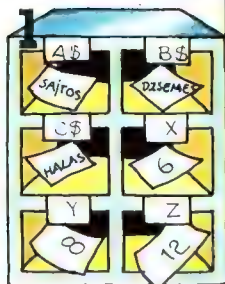
A \* szorzást jelent, a / osztást, az SQR pedig gyökvonást.



Most a **PRINT**-tel azt mondtuk a gépnek, hogy végezze el a műveleteket. Az összeadáshoz és a kivonáshoz a szokásos jeleket használjuk, a szorzáshoz a \*, az osztáshoz a / jeleket.

A számítógép összetettebb matematikai műveleteket is elvégez, pl. szögfüggvényeket számol, gyököt von stb

# Még többet a változókról



2

Üres helyek

PRINT "MEGESZEM ";X;" VAJAS KENYERET ÉS A ";A\$;" SZENDVICSEKET"  
MEGESZEM 6 VAJAS KENYERET ÉS A SAJTOS SZENDVICSEKET"

PRINT "MEGESZEM ";Z;" VAJAS KENYERET ÉS A ";C\$;" SZENDVICSEKET"  
MEGESZEM 12 VAJAS KENYERET ÉS A HALAS SZENDVICSEKET



A legtöbb számítógépnél az idézőjelek belül a szóközőket előre ki kell hagyni.

Nem mindig lehet megérteni, ha a változókat csak önmagukban íratjuk ki. Általában szükség van egy kis magyarázó szövegre. Ilyenkor szövegek és változók követik egymást. A változó mindkét oldalára tegyél pontosvesszőt, de hogy

meglegyenek a szóközők, az idézőjeleken belül hagyd ki egy üres helyet, ahogy a képen látod. Használhatsz vesszőt is, úgy viszont túl messze kerülhetnek a szavak!

3

LET X=X+1  
LET C\$="TON"+C\$



A program futása közben is változtathatsz a változók tartalmát, ahogy a képen is látható. Ezek az utasítások azt jelentik a gépnek, hogy legyen az **X** címkén egygel nagyobb szám, ill, hogy legyen a **C\$**-t jelölő szó ezentúl: tonhalas.

4

Üres helyek

PRINT "MEGESZEM ";X;" VAJAS KENYERET ÉS A ";C\$;" SZENDVICSEKET"

MEGESZEM 7 VAJAS KENYERET ÉS A TONHALAS SZENDVICSEKET

Amikor legközelebb megkérde a gépet, hogy írja ki a memóriarekeszeiből a változókat, már az új szavakat és számokat fogja kiírni.

5

10 LET A=9

20 LET B=7

30 PRINT A\*B

40 PRINT A/B

50 END

RUN

63

1.28571

Szorzás

Osztás

Változókat is összegezhetsz, ahogy a fenti programban látod. A gép a memóriában megtalálja a számokat, és elvégzi a műveleteket.

## Programfejtörők

1. Bővítsd ki a bal oldali programot úgy, hogy bizonyos számokat hozzáadsz a változókhoz! Az egyik eredmény 100, a másik 1 legyen, és egy sorban egy betűközzel jelenjenek meg.

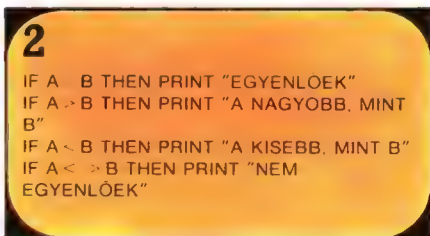
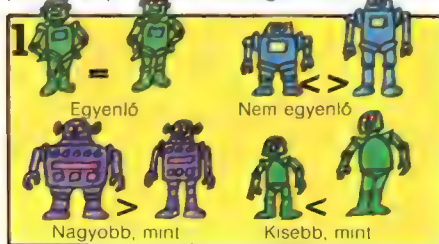
2. Változtasd meg a 30-as és a 40-es sort úgy, hogy szövegesen jelenjék meg a válasz "7-szer 9 az 63"!

3. A 15. oldalon levő programfejtörőt oldd meg úgy, hogy a gép a nevedet és az üzenetet egy sorba írja ki!



# Hogyan mérleget a számítógép?

Az egyik leghasznosabb dolog, amit a számítógép tud, hogy képes összehasonlítani adatokat, az eredménytől függően pedig mást és mást csinálni. Erre az IF (HA)... és a THEN (AKKOR) szavak szolgálnak.



A számítógép egy csomó összehasonlító vizsgálatnak tudja alávetni az adatokat. Ellendörzini tudja, hogy két adat egyenlő-e, különböző-e, ill. hogy egyik nagyobb vagy kisebb-e a másiknál.

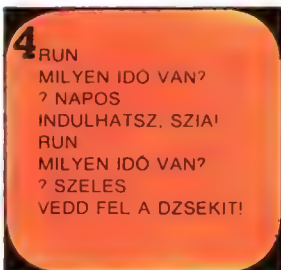
Ezek a sorok mutatják az IF és THEN használatát, amikor a számítógép összehasonlít két adatot. Mindenféle adatot összehasonlíthat; nemcsak számokat, hanem szavakat, sőt változókat is.

## 3 A gondoskodó program

```
10 PRINT "MILYEN IDŐ VAN?"
20 INPUT I$
30 IF I$ = "SZELES" THEN PRINT "VEDD FEL
A DZSEKIT"
40 IF I$ = "NAPOS" THEN PRINT "INDULHATSZ,
SZIA"
50 END
```



Íme egy program az IF és THEN használatára. A 20-as sorban a gép az I\$ változóba betett szót elraktározza. Ezután a 30-as és a 40-es sorban ellenőrzi, hogy a szó megegyezik-e a "szeles" vagy a "napos" szavakkal. Ha megegyezik,



kiírja az egyik választ. Ha a 20-as sorra mást válaszolsz, semmi sem fog történni. Ha viszont megváltoztatod a szavakat a 30-as és a 40-es sorokban, akkor már arra fog választ adni.

## Életkorprogram

```
10 PRINT "HÁNY ÉVES VAGY?"
20 INPUT A
30 IF A > 16 THEN PRINT "TÚL ÖREG"
40 IF A < 16 THEN PRINT "TÚL FIATAL"
50 IF A = 16 THEN PRINT "ÉPP A LEGJOBB"
RUN
HÁNY ÉVES VAGY?
? 16
ÉPP A LEGJOBB
```

## Angol lecke

```
10 PRINT "HOGY MONDOD ANGOLUL: PIROS?"
20 INPUT A$
30 IF A$ = "RED" THEN PRINT "HELYES"
40 IF A$ < > "RED" THEN PRINT
"NEM":A$;"HANEM RED"
RUN
HOGY MONDOD ANGOLUL: PIROS
? BLUE
NEM BLUE, HANEM RED
```

Az életkorprogramban a számítógép összehasonlítja a beadott A értéket a 16-os számmal. Kisebb érték esetén "túl fiatal", nagyobbánál "túl öreg" az illető, a 16 éves a "legjobb"

A másik programban a gép a két különböző válasz közül az egyiket biztos kiírja, aszerint, hogy az A\$ egyenlő-e a "red"-del vagy sem.

**Programlejtő** — Tudsz-e olyan programot írni, amelyben a számítógép összehasonlításokat kérdez tőled, és a válasza helyes, rossz válasza pedig megadja az eredményt?

## Programelágazások

```
1 IF A=6 THEN LET A$="HAT"  
  IF X=Y-2 THEN LET Z=0  
  
  IF S=T THEN STOP  
  
  IF R<10 GOTO 30
```

Ez azt mondja  
a gépnek, hogy menjen  
a 30-as sorra.



A THEN szó után jóformán bármilyen utasítást adhatsz a számítógépnek, ahogyan a képen is látod. Sokszor okos dolog pl. egy másik sorra küldeni. (A legtöbb számítógépnél kihagyhatod

```
2 10 INPUT K$  
  20 IF K$="IGEN" THEN GOTO 100  
  30 IF K$="NEM" THEN GOTO 200  
  
  100 PRINT "AZT ÍRTAD BE, HOGY IGEN"  
  110 STOP  
  
  200 PRINT "AZT ÍRTAD BE, HOGY NEM"  
  210 END
```

Ebben a két sorban  
szétágazik a prog-  
ram. (GOTO-MENJ)



a GOTO szót; a Sinclair gépeknél nem.)  
A GOTO-val végtelen sokáig futó programok is keletkezhetnek. Ezért szükség lehet a STOP utasításra.

## Matekprogram

```
10 PRINT "ÍRJ BE EGY SZÁMOT!"  
20 INPUT A  
30 PRINT "MOST EGY MÁSIKAT!"  
40 INPUT B  
50 PRINT "ÖSSZEADNI, KIVONNI,  
  SZOROZNI,"  
60 PRINT "OSZTANI VAGY LEÁLLNI"  
65 PRINT "AKARSZ?"  
70 INPUT C$  
80 IF C$="ÖSSZEADNI" THEN PRINT A+B  
90 IF C$="KIVONNI" THEN PRINT A-B  
100 IF C$="SZOROZNI" THEN PRINT A*B  
110 IF C$="OSZTANI" THEN PRINT A/B  
120 IF C$="LEÁLLNI" THEN STOP  
130 GOTO 10
```

```
RUN  
ÍRJ BE EGY SZÁMOT!  
? 17  
MOST EGY MÁSIKAT  
? 184  
ÖSSZEADNI, KIVONNI, SZOROZNI,  
OSZTANI VAGY LEÁLLNI  
AKARSZ?  
? ÖSSZEADNI  
201 ←  
ÍRJ BE EGY SZÁMOT!  
?
```

A PROGRAM CSAK  
AKKOR FOG LEÁLLNI, HA  
A LEÁLLNI-T  
VÁLASZTOD



A számítógép  
válasza

Ebben a programban a begépett számok az A-ban és a B-ben raktározódnak el, a választásod pedig C\$-ban. A 80-astól a 120-as sorig a gép összehasonlítja a C\$-t az öt különböző szóval, és ha valamelyikkel egyezik, végrehajtja az utasítást. Egyszerűen átsiklik azok felett a sorok felett, amelyek nem teljesülnek.

## Életkor-kitaláló program

```
1 10 PRINT "TALÁLD KI,  
  HÁNY ÉVES VAGYOK!"  
20 INPUT G  
30 IF G< > 14 THEN  
  PRINT "PRÓBÁLD MEG  
  ÚJRA!"  
40 IF G< > 14 THEN GOTO 20  
50 PRINT "HELYES"  
60 END
```

```
2 RUN  
  
TALÁLD KI,  
HÁNY ÉVES VAGYOK!  
? 15  
PRÓBÁLD MEG ÚJRA!  
? 14  
HELYES
```

```
3 TALÁLD KI,  
  HÁNY ÉVES VAGYOK!  
? 15  
FIATALABB  
? 13  
IDŐSEBB  
? 14  
HELYES
```

HOGYAN ÍR-NÁD  
MEG A  
PROGRAMJÁT?



Ez a program addig ismétlődik, amíg G=14 nem lesz. Amikor végre G=14, a számítógép átsiklik a 30-as és a 40-es

sorokon, és azt írja ki, hogy "helyes". Próbáld kiegészíteni a programot, hogy könnyebb legyen kitalálni a számot!



# Egy kicsit komolyabb programok

Ezen a két lapon a programok majdnem mindent felhasználnak a BASIC-ből, amit eddig tanultunk. Az első program egy kétszemélyes űrjáték. Ha nincs géped, akkor is nézd át a programokat, és próbáld következtetni, hogyan működnek!

## Őrjárat az űrben

```
10 PRINT "AZ IDEGEN NÉGYZET  
JOBBRA"
```

```
20 INPUT A
```

```
30 PRINT "AZ IDEGEN NÉGYZET  
FELFELÉ"
```

```
40 INPUT B
```

```
50 CLS
```

```
60 PRINT "AZ ŐRJÁRAT NÉGYZETE JOBBRA"
```

```
70 INPUT C
```

```
80 PRINT "AZ ŐRJÁRAT NÉGYZETE FELFELÉ"
```

```
90 INPUT D
```

```
100 CLS
```

```
110 LET X = SQR((A - C)*(A - C) + (B - D)*(B - D))
```

```
120 PRINT "MOST ÉPP"
```

```
130 PRINT X;"ÜREGSÉGNYI TAVOLTSÁGBAN  
VAGYUNK, TEHÁT"
```

```
140 IF X < 1.5 THEN PRINT  
"AZ IDEGENT ELFOGTUK"
```

```
150 IF X < 1.5 THEN STOP
```

```
155 PRINT "AZ ŐRJÁRAT ÚJ HELYE"
```

```
160 GOTO 10
```

```
170 END
```

A 10-es-től a 40-es sorig A-ban és B-ben elraktározzuk az idegen koordinátáit

Az 50-es sor eltünteti a képernyőről az idegen koordinátáit

A 60-as-tól a 90-es sorig C-ben és D-ben elraktározzuk az őrszem koordinátáit

Ez a sor kiszámítja a távolságukat, és az eredményt X-ben elraktározza

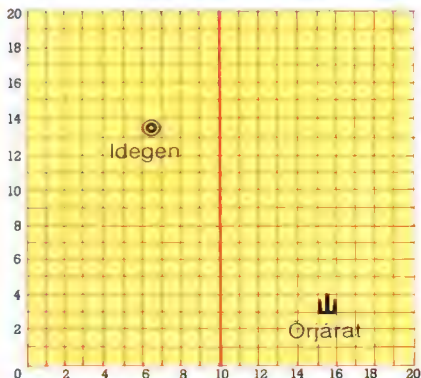
Ha X kisebb, mint 1.5, vége a játéknak, ha nagyobb, kezdődik előlről

Ebben a játékban az egyik játékos egy ellenséges idegen, akit az őrjárat megpróbál elkapni. Mindkét játékos egy titkos térképet rajzol, amelyen beméri a helyzetét (hogy hogyan, azt a rajzon majd megmagyarázzuk). Megadják

a gépnek a helyzetük hálózati koordinátáit, amiből a gép kiszámítja a távolságukat. A játékosok a gép válasza alapján változtatják tovább a helyüket.

## Hogyan játsszuk le?

A titkos térképhez mindegyik játékos egy  $20 \times 20$ -as négyzethálót rajzol, és ráírja a számokat, amint az a képen látható. Az idegen a háló bal oldaláról indul, az őrjárat pedig jobbról. Mindegyik fordulóban két kockányit léphetnek felfelé, lefelé, oldalra vagy átlósan, és azután megadják a számítógépnek az új helyzetüket. Ha másfél kockánál közelebb kerülnek egymáshoz, az őrjárat elkapta az idegent.



# Mit csináljunk, hogy a számítógép okosnak tűnjön?

Ebben a programban a gép bebizonyítja, hogy még beszélgetni is tud. A lap alján levő képek egy-egy lehetséges társalgást mutatnak be. A program újfajta módon használja az INPUT utasítást, ez a módszer gazdaságosabb és jobban is olvasható.

1

```
10 INPUT "KÉREK EGY SZÁMOT";N
20 INPUT "MOST EGY MÁSIKAT IS?";M
30 PRINT N;"",M;" AZ"N*M
```

A pontosvessző nem minden gép esetében kell



2

A ZX 81-en ezt kell beírnod:  
10 PRINT "KÉREK EGY SZÁMOT"  
15 INPUT N

```
RUN
KÉREK EGY SZÁMOT ? 10
MOST EGY MÁSIKAT IS ? 8
10 X 8 AZ 80
```



A legtöbb számítógépen (a ZX 81-en nem) az INPUT sort érthetőbbé tehetjük, ha idézőjelben a változó elé írjuk a jelentését.

Amikor lefuttatod a programot, e szavak után megjelenik az INPUT kérdőjele.

## A program

```
5 LET C=0
10 PRINT "SZERETNÉK BESZÉLGETNI VELED,"
20 INPUT "MESElj VALAMI VICCESET,
    AMI MOSTANÁBAN TÖRTÉNT VELED!"; A$
30 READ B$
40 PRINT B$
50 INPUT C$
60 LET C=C+1
70 IF C=6 THEN GOTO 100
80 GOTO 30
90 DATA "MIÉRT?", "EZ MIÉRT TÖRTÉNT?"
95 DATA "MIÉRT", "MEG TUDOD MAGYARÁZNI?"
98 DATA "MEG TUDOD MONDANI MIÉRT," "MI VOLT AZ OKA?"
100 PRINT "AZT MÉSÉLTED"
110 PRINT " ";A$
120 PRINT "DE MAGAD IS ELISMERED:"
130 PRINT " "; C$
140 PRINT "FURCSÁ EGY ESET."
150 PRINT "BESZÉLGESSÜNK MÉG! FUTTASS LE ÚJRA!"
160 END
```

Ez a bevétel újfajta módja. A válaszod az A\$-ban van elraktározva.

A 30-as sorban a számítógép kikeresi az első DATA sort, veszi az első adatot, és beteszi B\$-ba

A 60-as és a 70-es sorokban a C változó egy számláló. Azt számolja, hogy hányszor ismétlődik a program. Ha C=6, mindegyik DATA egységet feldolgoztuk, és a számítógép továbbmegy a 100-as sorra.

A 80-as sor visszaküld a 30-asra, és az adattalista következő szava kerül B\$-ba.

110-es és 130-as sorok szóközei egy kis helyet hagynak a képernyőn a válaszod előtt. Egyébként nincs különösebb jelentősége, hogy a programba hány szóközt gépeltél be.

## Hogyan működik?

1

RUN  
SZERETNÉK  
BESZÉLGETNI VELED.  
MESElj VALAMI  
VICCESET. AMI  
MOSTANÁBAN TÖRTÉNT  
VELED!

BELEESTEM EGY GÖDÖRBE

2

MIÉRT?

NEM  
NÉZTEM, HOGY  
HOVA' LEPEK

3

EZ MIÉRT TÖRTÉNT?

FAGYALTOT  
ETTEM

MEG TUDOD MAGYARÁZNI?

4

LENYALTAM  
AZ ÚJJAIMRÓL

MIÉRT?

MERT  
OLVADT

MEG TUDOD  
MONDANI. MIÉRT?

5

MERT DUGDOSTAM

MI VOLT  
AZ OKA?

NEM  
AKARTAM, HOGY A  
BARÁTOM MEGLÁSSA,  
FAGYALT VAN NÁLAM

AZT MÉSÉLTED.  
BELEESTEM EGY GÖDÖRBE.  
DE MAGAD IS ELISMERED:  
NEM AKARTAM, HOGY  
A BARÁTOM MEGLÁSSA,  
FAGYALT VAN NÁLAM  
FURCSÁ EGY ESET  
BESZÉLGESSÜNK MÉG! FUTTASS  
LE ÚJRA!



# Rajzoljunk képeket!

A számítógépes képek sok kicsi, kivilágított pontból tevődnek össze. Ezeket a négyszög alakú pontokat képpontoknak hívjuk.

A pontocskák kigyújtására külön utasítás szolgál. A legtöbb gép még színezní is tudja a pontokat, de itt a színezéssel nem foglalkozunk. Ezen a két oldalon egyszerű képeket rajzoló BASIC utasításokat mutatunk be.

2

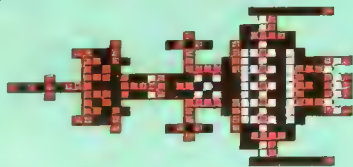
Íme néhány szokásos utasítás a képpontok kigyújtására.



PLOT (20, 30)\*  
SET 20, 30 vagy  
PLOT 69, 20, 30

Ez az utasítás gépenként eltérő, de lényegében mindegyik a PLOT(X, Y)-hoz hasonlatos. (X, Y) adja meg a képpont helyét a képernyőn: jobbra az X-edik, felfelé az Y-adik képpontot jelenti.

1



Mindegyik számítógép képpontokból építi fel a képet, csak hogy az egyiknél jól elkülöníthetőek a négyszögek, a másiknál alig észrevehetően aprók. Ehhez a finom felbontáshoz sok ezer pont kell, ami rengeteg adatot jelent, és azért nagy memóriára van szükség.

3



Egy finom felbontású képernyőn akár 1000 x 1000 pontot berajzolhatsz. A szerényebb gépeknél csak kb. 60 x 40 van. (Ha túl nagy számot helyettesítesz X-be vagy Y-ba, akkor "lelépsz" a képernyőről, és a gép hibaüzenetet ad!)

4



Ne felejtse el: némelyik számítógépnél át kell térni a grafikus üzemmódra!

5



6

UNPLOT (20, 30).  
RESET 20, 30



A számítógépes képrajzolást általában grafikának hívjuk. Bizonyos gépeknél egy különleges utasításra van szükség, amivel áttérhetünk a grafikus üzemmódra.

Ezek az utasítások kioltják a fénylő pontokat. A sokféle utasítástajtból mi a PLOT (X, Y)-t és az UNPLOT (X, Y)-t fogjuk használni. A programok kipróbálásánál természetesen a géped saját szavát kell begépeined.

\* A PLOT jelentése magyarul: berajzol.

# Rajzolóprogram

1

```
10 PRINT
  "ÍRJ BE KÉT SZÁMOT"
20 INPUT X
30 INPUT Y
40 PLOT (X, Y)
50 GOTO 10
```

Lehet, hogy a PLOT helyett mást ért meg a géped!



Ne felejtssd el a számok után megnyomni a **NEWLINE**-t vagy a **RETURN**-t!

2

```
RUN
ÍRJ BE KÉT SZÁMOT
? 24
? 24
ÍRJ BE KÉT SZÁMOT
? 30
? 15
```

← Első képpont

← Második képpont



Ez a rövid program két számot kér, és ezekkel a számokkal mint koordinátákkal berajzolja a képpontot. Légy óvatos az X, Y értékekkel, nehogy egy képernyőn kívüli pontot akarj kigyujtani!

## Képet rajzolunk

```
10 LET X = 10
20 LET Y = 10
30 PLOT (X, Y)
40 LET X = X + 1
50 LET Y = Y - 1
60 IF X < 14 THEN GOTO 30
```

Ez egy lefelé, menő átlós vonalat rajzol:

```
100 LET Y = Y + 1
110 X = X + 1
120 PLOT (X, Y)
130 IF X < 20 THEN GOTO 100
```

Ez egy felfelé, menő átlós vonalat rajzol

Ha X-hez 1-et adunk és Y-hoz semmit, akkor vízszintes vonalat kapunk.

Ha Y-hoz adunk 1-et és X-hez semmit, akkor függőleges lesz a vonal.

Először mindig kockás papírra rajzold le a képet, és azon számítsd ki a koordinátákat.

Ezután következik a program kidolgozása. Keresd meg az alkalmas kezdőpont értékeit, és hozzáadogatással vagy kivonogatással rajzoltasd meg a képpontok sorozatát, ahogy a képen is látod!



Egy kész képet könnyű módosíthatni, csak a számokat kell megváltoztatnod a programban. Ha a kezdőértéket változtatod, a teljes kép elmozdul, ha a koordinátákat megszorozod mondjuk 3-mal, mint ha szétrobbantanád a képet. (Próbáld ki más számok hatását is!)

## Egy másik módszer



A képernyőgrafikának van egy egészen más módja is, amivel nem kell se számolgatni, se programozni. Kiegészítheted a gépedet egy grafikus táblával. Ha erre ráhelyezünk egy rajzot és egy külön kis készülékkel (a manóval) végigkövetjük, akkor a képpontok koordinátái automatikusan a gép memóriájába kerülnek.



# Játsszunk véletlen játékokat!

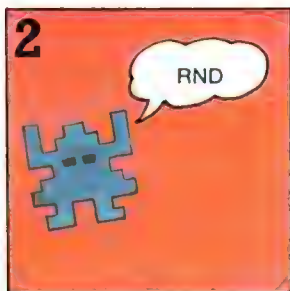
1



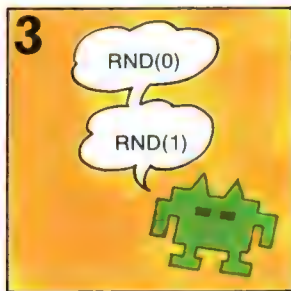
Amikor kockát dobtok, nem lehet tudni, melyik szám kerül felülre. Egytől hatig bármelyik szoba jöhet. A számítógép is elő tud állítani megjósolhatatlan számokat: ezeket véletlen számoknak hívjuk.

A véletlen számok előállítására egy különleges program van a gépben. Megeshet, hogy ugyanaz a szám többször egymás után megismétlődik – akárcsak a kockadobásakor –, de nagyon sok véletlen számot kipróbálva mindegyik nagyjából egyforma sokszor fordul elő.

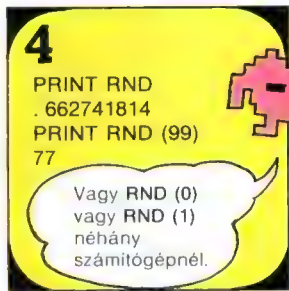
2



3

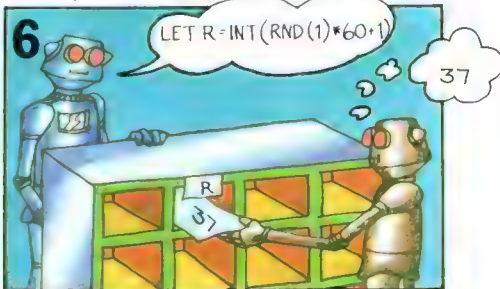
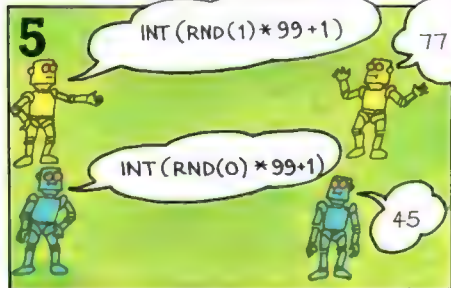


4



A véletlen számok előhívásához az **RND** (RaNDom; véletlen) kulcsszó szükséges. Bizonyos gépeken az **RND** után még zárójelben egy 1-eset vagy egy 0-át is kell írni. (Ellenőrizd a gépkönyvben, hogy a géped melyik alakot használja!)

A számítógép mindig 1-nél kisebb (de nullánál nem kisebb!) számot állít elő, viszonylag egyszerű számítással véletlen egész számokat is létre tudunk hozni. Egyes gépeknél elég, ha a lehetséges legnagyobb számot zárójelben az **RND** után írjuk. Pl. **RND (99)**.



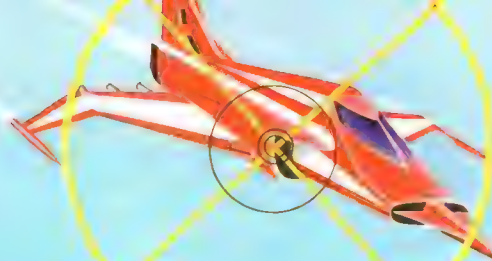
Más gépeknél, ahol átszámítás szükséges a véletlen egész számokhoz, az **INT** (INTEger = egész) szót is fel kell használnunk. Itt a szorzótényező a legnagyobb lehetséges szám, és ha az 1 a legkisebb szoba jövő érték, még 1-et hozzá is kell adni.

Ez az utasítás kiválaszt egy véletlen számot és az **R** változóban elraktározza. Mi ezt a fajta kifejezést fogjuk használni, de te feltétlenül ellenőrizd a gépkönyvben! Az **INT (RND (1)\*60 + 1)** kifejezés arra utal, hogy a kiválasztott véletlen szám 1 és 60 közötti egész.



# Úrtámadás

Ebben a játékprogramban a véletlen számoknak is van szerepük. Azt játsszuk, hogy egy űrhajón vagy, amelyet ellenséges vadászgépek hulláma támadott meg. Az űrhajód számítógépe beméri az ellenséget és kiírja a koordinátáit. Egy-egy ellenséget úgy lehet kilőni, ha begépeled koordinátáinak a szorzatát



```
10 LET C=0
20 LET A=INT (RND (1)*20 + 1)
30 LET B=INT (RND (1)*20 + 1)
40 PRINT "AZ ELLENSÉGES HAJÓ KÓDJAI"
45 PRINT A,B;"TÜZ"
50 INPUT X
60 LET C=C+1
70 IF X =A*B THEN PRINT
   "EGY ELLENSÉGGEL KEVESEBB"
80 IF X < > A*B THEN PRINT "NEM TALÁLT"
90 IF C<6 THEN GOTO 20
100 END
```

A C számláló számon tartja, hogy hányszor ismétlődött a program. A 60-as sor minden alkalommal 1-et ad C-hez.

Ez a két sor az ellenséges űrhajó véletlen koordinátáit állítja elő az A, ill. a B változóban.

A te számod az X-ben van elraktározva. A 70-es és a 80-as sorban a gép ellenőrzi, hogy a helyes választ kapta-e.

Ez a sor újra végrehajtja a programot, ha C<6.

## A program lefuttatása

A jobb oldali kép mutatja a játék menetét. Ha a helyes szorzatot adod meg, a gép kiírja, hogy "egy ellenséggel kevesebb". Ha válaszod nem egyenlő A\*B-vel, "nem talált" visszajelzést kapsz.

```
RUN
AZ ELLENSÉGES HAJÓ KÓDJAI
17 3 TÜZ
? 41
NEM TALÁLT
AZ ELLENSÉGES HAJÓ KÓDJAI
11 5 TÜZ
? 55
EGY ELLENSÉGGEL KEVESEBB
AZ ELLENSÉGES HAJÓ KÓDJAI
13 6 TÜZ
```

A 45 sorban levő vessző miatt a számok elkülönítve jelennek meg.

## Programfejlesztő

Ennek a programnak van egy gyengéje, mégpedig az, hogy éppen a találatokat nem számolja. Legyen a találatsszámláló S, amelyet a program elején 0-ra állítasz, és találatonként eggyel novelsz, a játék végén pedig kiíratod az értékét.

### Véletlen minta rajzolása

```
5 CLS
10 LET X=INT (RND (1)*30 + 1)
20 LET Y=INT (RND (1)*30 + 1)
30 PLOT (X,Y)
40 GOTO 10
```

Ezzel letöröljük a képernyőt, mielőtt a képpontok megjelennek.

A véletlen számoknak rá kell léniük a képernyőre!

Ez a sor végteleníti a programot

A véletlen számokat felhasználva találomra pontokat rajzolunk ki a képernyőn. A 10-es és a 20-as sor 1 és 30 közötti véletlen számokat gyárt, ezekből lesz a képpont X és

Y koordinátája a 30-as sorban. A pötytyözés eleinte gyorsan megy, később már ritkábban gyullad fel még egy-egy hiányzó pont. A program csak BREAK-kel állítható le.

A CLS, az RND és a PLOT eltérő lehet a gépeden, esetleg grafikai üzemmódra is át kell állnod!





# Készítsünk ciklusokat!

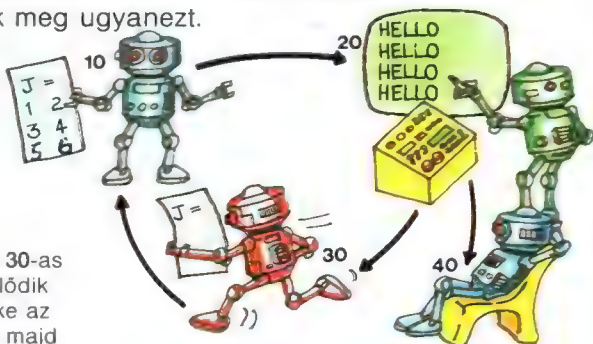
A géppel gyakran ugyanazokat a lépéseket többször is el kell végeztetned a programon belül. A 21. oldalon a **GOTO** utasítással és egy számláló beállításával oldottuk meg az ismételtetést. Van azonban egy másik módszer is, amelyben a **FOR...TO...** és a **NEXT** ciklusutasításokkal valósítjuk meg ugyanezt.

## 1 Helló ciklus

```
10 FOR J=1 TO 6
20 PRINT "HELLO"
30 NEXT J
40 END
```

Ciklus

Ebben a programban a 10-es és a 30-as sor közötti ciklusban hatszor ismétlődik a 20-as sor. J a ciklusváltozó, értéke az első menetben 1, a másodikban 2, majd 3 és így tovább, egészen hatig. A 20-as sor kiírja a **HELLO**-t, a 30-as pedig visszaléptet a 10-es sorra, hogy vegye



a J következő értékét. Amikor a J eléri a 6-ot, a számítógép továbbmegy a 40. sorra.

## 2 Beugratós program

```
10 FOR J=1 TO 8
20 PRINT "2 MEG 2 AZ 5"
30 NEXT J
40 PRINT
50 PRINT "CSAK VICC VOLT!"
60 END
```

Ciklus

Néhány gépen nincs felkiáltójel, úgyhogy azt kihagyhatod.

```
2 MEG 2 AZ 5
2 MEG 2 AZ 5
2 MEG 2 AZ 5
2 MEG 2 AZ 5
2 MEG 2 AZ 5
2 MEG 2 AZ 5
2 MEG 2 AZ 5
2 MEG 2 AZ 5
CSAK VICC VOLT!
```

Ebben a programban a 10-es sor és a 30-as sor közötti ciklusban a 20. sor nyolcszor ismétlődik. Valahányszor átmegy a 20-as soron a program, ugyanazt

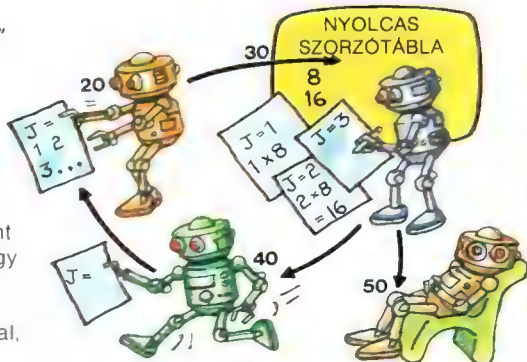
a hibás összeget írja ki. Amikor túl van a nyolcadik kiíráson, végrehajtja a program hátralevő részét. A 40-es sor utasítása csupán egy üres sort hagy.

## 3 Nyolcas szorzótábla

```
10 PRINT "NYOLCAS SZORZÓTÁBLA"
20 FOR J=1 TO 12
30 PRINT J*8
40 NEXT J
50 END
```

Ciklus

Ez alkalommal a J-t nemcsak ciklusváltozóként használjuk, hanem egyúttal a  $J \cdot 8$  szorzat egyik tényezőjeként is. A 20-as sor azt mondja a gépnek, hogy legyen a J először 1, majd 2, 3 stb. egészen 12-ig. A 30-as sor veszi a J mindenkoros értékét, megszorozza 8-cal, és kiírja a választ. Ezután a 40-es sor visszaküldi a gépet a 20-as sorra, hogy vegye a következő J értéket.



## Készítsünk mintát!

A **FOR...NEXT** ciklusok arra is alkalmasak, hogy segítségükkel mintákat készítsünk. Itt pl. egy egyszerű alakzatot ismételgetünk. Ennek a mintának a teljes programját túlságosan hosszú lenne itt leírni, de valahogy így nézne ki:

```
10 FOR I = 1 TO 45
20 Rajzolj egy négyszöget, és
   minden alkalommal egy kicsit
   módosítsd a helyzetét!
30 NEXT I
40 END
```

## Lépések

Előfordulhat, hogy a **J** értékét 1-től különböző értékkel kell léptetned. Pl. hármasával vagy hetesével. Ilyenkor még a **STEP (LÉPÉS)** szót is be kell vened a ciklusba. A következő programban a **STEP - 1** hatására minden alkalommal, amikor a gép átmegy a 10-es és a 40-es sor közötti cikluson, **J** értéke eggyel csökken.

## A falánkok programja

```
5 CLS
10 FOR J = 7 TO 2 STEP - 1
20 PRINT "MÉG „J,” SÜTEMÉNY VAN"
30 NEXT J
40 PRINT
50 PRINT "MINDJÁRT SZÉTPUKKADOK"
60 FOR K = 1 TO 1000
70 REM: NE CSINÁLJ SEMMIT
80 NEXT K
90 PRINT
100 PRINT "DURR"
```

A **J = 2**-nel befejeződik a ciklus (2 sütemény marad).

Ciklus

Ciklus

MÉG 7 SÜTEMÉNY VAN  
MÉG 6 SÜTEMÉNY VAN  
MÉG 5 SÜTEMÉNY VAN  
MÉG 4 SÜTEMÉNY VAN  
MÉG 3 SÜTEMÉNY VAN  
MÉG 2 SÜTEMÉNY VAN

MINDJÁRT  
SZÉTPUKKADOK  
DURR

A gépek számolási sebessége változó, lehet, hogy kisebb számra van szüksége a 60-as sorban, pl. 500-ra vagy 250-re.



Ebben a programban két ciklus van. Az első ciklus hatszor iratja ki a sütemények számát. A **J** értéke minden alkalommal eggyel csökken, 7-től 2-ig. A második ciklus magában számlál 1-től 1000-ig, és látszólag semmi sem történik. A gép kb. 1 perc alatt fut végig a számokon, és

ennek kb. egy perc szünet az eredménye. A **REM**-mel (**REMark** : megjegyzés) kezdődő sorokat a számítógép nem veszi figyelembe, arra valók, hogy később emlékeztessenek, mit is csinál egy-egy programrészlet.

## Programfejlesztő

1. Tudod úgy módosítani a 8-as szorzótáblát, hogy a "**J\*8**" is megjelenjen?
2. Próbáld egy általános szorzótáblaprogramot készíteni! Ez a program tetszőleges beadott **N** számnak kiírja a szorzótábláját.

Először a gép **N**-et kérje, majd egy ciklusban jelenjenek meg a számjegyek. Beiktathatsz még néhány sort a végére is: megkérdezheti a program, hogy készítsen-e másik szorzótáblát, és ha igen, léptesd vissza a kezdősorra!



# Ciklusok trükkjei

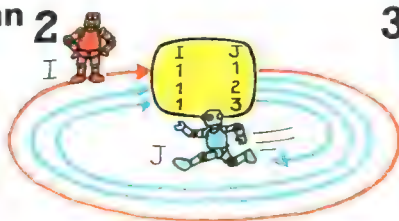
Most még mindig a ciklusokkal foglalkozunk. Megmutatjuk, hogy hogyan lehet a ciklusokon belül is ciklust szervezni. Ezeket egymásba ágyazott ciklusoknak nevezzük.

## 1 Ciklus a ciklusban 2

```

5 PRINT "I", "J"
10 FOR I = 1 TO 3
20 FOR J = 1 TO 3
30 PRINT I, J
40 NEXT J
50 NEXT I
60 END
    
```

I ciklus  
J ciklus



I	J
1	1
1	2
1	3
2	1
2	2
2	3
3	1
3	2
3	3

Azt mondhatjuk, hogy itt a J ciklus be van ágyazva az I ciklusba. Valahányszor a program belép az I ciklusba, 3-szor végigfut a J cikluson is. A ciklusmag,

vagyis a 30-as sor kiírja a pillanatnyi I és J értéket. A képek az egymásba épített ciklusokat, ill. a program eredményét mutatják.

## Számítógépes óra

5 CLS

10 FOR P = 0 TO 59

20 FOR M = 0 TO 59

30 PRINT P;"":M

40 CLS

50 NEXT M

60 NEXT S

70 END

A másodpercek  
ciklusa  
A percek  
ciklusa

0:45

Az idő beállításához  
használt ezt  
a késleltetőciklust:  
54 FOR Z = 1 TO 100  
58 NEXT Z



## Ciklushibák

```

10 FOR I = 1 TO 4
20 FOR J = 1 TO 4
30 PRINT I
40 PRINT J
50 NEXT I
60 NEXT J
    
```

A beépített ciklus összes  
utasítása a külső  
cikluson belül kell,  
hogy legyen.

A számítógép belsejében egy elektronikus óra szabályozza a gép valamennyi működésének a ritmusát. Az óra másodpercenként egy – négy milliószer ketyeg. A fenti program hatására a gép úgy fog viselkedni, mint egy digitális óra. Egymásba ágyazott ciklusok számlálják a másodperceket, ill. a perceket.

A másodpercek ciklusát minden percciklusban 60-szor hajtja végre a program. Első kipróbáláskor az óra egész biztosan sietni fog. Ezért külön be kell építened egy késleltetőciklust, amelynek a végértékét próbálgatással tudod beállítani úgy, hogy számítógépes órad pontosan járjon.

## Véletlenszám-ellenőrző

```

10 FOR I = 1 TO 1000
20 LET R = INT (RND (1) * 6 + 1)
30 IF R = 1 THEN LET A = A + 1
40 IF R = 2 THEN LET B = B + 1
50 IF R = 3 THEN LET C = C + 1
60 IF R = 4 THEN LET D = D + 1
70 IF R = 5 THEN LET E = E + 1
80 IF R = 6 THEN LET F = F + 1
90 NEXT I
100 PRINT "VÉGE"
110 PRINT A, B, C
120 PRINT D, E, F
130 END
    
```



Ha unod a hosszú várakozást, a 10-es sorbeli számot 500-ra vagy 250-re változtathatod, hogy hamarabb véget érjen a program.

RUN  
VÉGE

162	168	167
160	187	156

Azt mondtuk a gép véletlen számainról, hogy nagyjából egyforma eséllyel fordulnak elő. Most ezt ellenőrizzük. A program 1000 darab számot hoz létre egy és hat között és az A, B, C, ..., F változókban számolja, hány 1-es, 2-es, ..., 6-os fordul elő, majd a végén kiírja az eredményeket. •

# Mintaismétlő program

Ez a program egymásba ágyazott ciklusok segítségével egy kis mintát ismétel az egész képernyőn. A program nagyon bonyolultnak látszik, de ha gondosan átolvasod, és végiggondolod, hogy az egyes sorok mit csinálnak, hamarosan megérted a működését. A minta alakját véletlen számok határozzák meg, így a különböző futtatások során mindig másmilyen mintákkal tölti be a képernyőt.

Finom felbontású grafika esetében használj nagyobb véletlen számokat! (Kb. ezek 10-szeresét!)



```

5 CLS
10 LET A = INT (RND (1)*6 + 1)
20 LET B = INT (RND (1)*7 + 1)
30 LET C = INT (RND (1)*6 + 1)
40 LET D = INT (RND (1)*4 + 1)
50 INPUT "HÁNY PONT SZÉLESSÉGÜ A KÉPERNYŐ";W
60 INPUT "HÁNY PONT MAGASSÁGÜ A KÉPERNYŐ";V
65 CLS
70 FOR I = 0 TO V - 1 STEP V/6
80 FOR J = 0 TO W - 1 STEP W/6
90 PLOT (J + A, I + B)
100 PLOT (J + A, I + C)
110 PLOT (J + C, I + D)
120 PLOT (J + B, I + D)
130 NEXT J
140 NEXT I
150 END
    
```

Ezek a sorok kiválasztják a véletlen számokat a mintához, és elraktározzák A-ban, B-ben, C-ben és D-ben

Az 50-es és a 60-as sor a képernyő (W) szélességére és (V) magasságra kérdez rá.

Az I ciklus megszámozza, hogy hányzor ismétlődik a minta felfelé a képernyőn. Az I lépésenként a képernyő (V) magasságának a hatodrészevel nő, s így a minta hatszor ismétlődik a képernyőn felfelé

Valahányszor a 90-tól a 120-as sorig jut a gép, I és J pillanatnyi értékehez hozzáadja a véletlen számokat, és ezekkel a koordinátákkal négy pontocskát rajzol ki

A J ciklus azt számolja, hányzor ismétlődik a minta vízszintesen. Ugyanúgy működik, ahogyan az I ciklus

## 1. Hogyan működik?



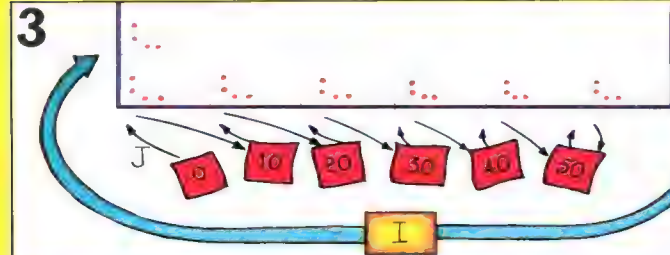
Képzeld meg, hogy a számítógép a 2, 5, 3 és 1 véletlen számokat választotta ki, és hogy a képernyő szélessége és magassága egyaránt 60.

## 2



Amikor először fut végig a program, az I és a J értéke 0, így a számítógép a pontokból alkotott első mintát tisztán a véletlen számok felhasználásával rajzolja ki. A 130-as sorból viszont visszaugrik a 70-es sorra; J második értéke J + 60/6, azaz 10 lesz. A következő mintában a koordináta már 10-zel nagyobb, és így tovább. A képernyő alsó sorában 6 minta jelenik meg.

## 3



Ha neked nem működik a program, próbáld ki újra, kisebb V és W értékekkel!

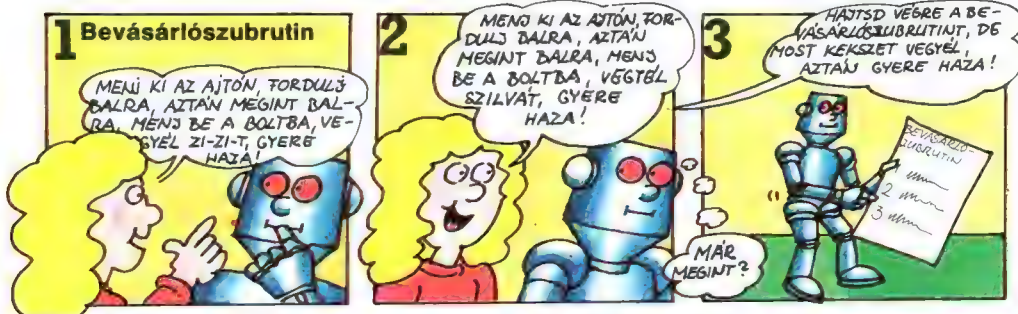


A J ciklus tehát hatszor ismétlődik, esetünkben J tízesével nő, és a négy képpont egy véletlen elrendezésében hatszor megjelenik a képernyő alján. Ezután I felveszi a követ-

kező értékét, a 10-et. J értéke először megint 0, és a gép kirajzolja a mintát a következő sorában I = 10-zel és 10-enként növekvő J-vel, ahogy az előbb.

# Szubrutinok

A szubrutinok afféle miniprogramok a nagyobb programokon belül. Valamilyen részfeladatokat hajtanak végre, mondjuk számokat adnak össze, vagy nyilvántartják az eredményeket, és ha bármikor szükséged van a részfeladat végrehajtására, csak oda kell irányítanod a programot. Ezzel megtakarítasz egy csomó programsort, nem kell többször leírni ugyanazt, rövidebb, sőt könnyebben áttekinthető is lesz a programod.



Képzeld el, hogy van egy programozható kis robotod, amit különböző feladatokkal el lehet szalajtani. Pl. el akarod küldeni vásárolni, ezért pontos utasításokkal látod el, hogy merre menjen. Igen ám, de

minden alkalommal, amikor a boltba küldöd, végig kell sorolnod ugyanazokat az utasításokat. Így aztán sokkal jobb, ha egy bevásárlószubrutint írsz, és esetenként arra hivatkozol.

## 4 Bevásárlóprogram



```

10 PRINT "MIRE VAN SZÜKSÉG A BOLTBÓL"
20 INPUT X$
30 GOSUB 100
40 PRINT "MÉG VALAMIT"
50 INPUT M$
60 IF M$ = "IGEN" THEN GOTO 10
70 STOP

100 REM: BEVÁSÁRLÓSZUBRUTIN
110 PRINT "MENJ EL, FORDULJ BALRA"
120 PRINT "ÚJRA BALRA, LÉPJ BE A BOLTBA"
130 PRINT "VEGYÉL "; X$; "GYERE HAZA!"
140 RETURN
    
```

Ki ne felejtse a RETURN sort!



A 30-as sor a gépet szubrutin első sorára küldi.

A főprogram után a STOP-pal le kell állítani a gépet, különben rátér a szubrutinra

Érdemes emlékeztetőül egy REM sorral kezdeni a szubrutint, így nem felejtet el, hogy mire való.

Ez az utasítás, RETURN (TÉRJ VISSZA) a GOSUB utáni 40-es sorra küldi vissza a programot

A BASIC-ben GOSUB utasításai hívhatsz meg egy szubrutint. A szubrutin végén a RETURN utasítással lehet visszatérni a főprogramra. A GOSUB után a szubrutin sorszáma áll; a RETURN után

nem kell szám, automatikusan folytatja a GOSUB utáni sortól a főprogramot. Egy programban annyi szubrutint alkalmazhatsz, amennyit csak akarsz.



## Szubrutinok

A szubrutinoknak olyankor vesszük hasznát, ha a programban különböző helyeken ugyanazt (vagy nagyon hasonló) feladatot akarunk végrehajtani. Íme még néhány példa a szubrutinok használatára!

### Osztást ellenőrző program

```
50 INPUT A
60 INPUT B
70 GOSUB 250
80 PRINT "A OSZTVA B-VEL = ";A/B
90 GOTO 50
250 REM: LEÁLLÍTÓ SZUBRUTIN
260 IF B=0 THEN STOP
270 RETURN
```

Köztudomású, hogy 0-val nem lehet osztani. Enélkül a szubrutin nélkül hibajelzéssel állna le a gép, így viszont még az osztás előtt a STOP utasítás állítja meg. Persze a leállás okát is kirathatnánk. A GOTO 50 miatt a szubrutin ele nem kell STOP.

### Sebességátváltó-program

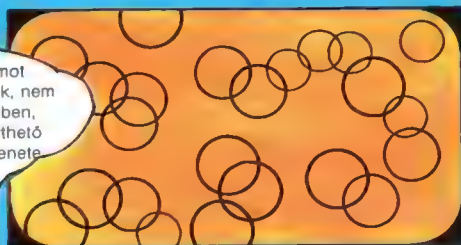
```
100 INPUT "TÁVOLSÁG";K
110 INPUT "IDŐ"; T
120 GOSUB 200
130 PRINT "AZ ÁTLAGSEBESSÉG"
140 PRINT K/T; "KM PER ÓRA"
    ÉS";M/T;"MÉRFÖLD PER ÓRA"
150 STOP
200 REM: KM-MÉRFÖLD ÁTVÁLTÓ
210 LET M=K/1.603
220 RETURN
```

Ez a szubrutin a kilométereket átváltja mérföldekre és km/óra-ban, ill. mérföld/óránban írja a sebességet. Az ilyen szubrutinoknak arra kell vigyazni, hogy ne keveredjenek a változók.

### Körprogram

```
1 A kör közepe=X,Y
2 A kör sugara=R
3 Szín=S
4 10-es szubrutin
5 Menj az 1-re!
10 Rem: Szubrutin a kör rajzolására
11 Rajzolj egy kört X,Y középponttal;
    R sugarúval és S színnel!
12 Return
```

Ezt a programot magyarul írtuk, nem pedig BASIC-ben, így jobban érthető a gondolatmenete.



Szubrutinokat grafikai programokban is alkalmazhatunk, úgy mint itt, ahol köröket

## Rejtvényprogram

```
5 LET C=0
10 PRINT "MIKOR TALÁLTÁK FEL EZEKET
    A DOLGOKAT?"
20 READ C$F ]
30 PRINT C$ ]
40 INPUT A
50 LET C=C+1
60 GOSUB 100
70 IF C=3 THEN STOP ]
80 PRINT "PRÓBÁLJ EGY MÁSIKAT"
90 GOTO 10
100 REM: VÁLASZOK SZUBRUTINJA
110 IF ABS(A-F)>10 THEN PRINT "IGEN"
120 IF ABS(A-F)<10 THEN PRINT "NEM"
130 RETURN
```

200 DATA TELEFON, 1876, NYOMDAGÉP, 1450, KERÉKPÁR,

1791

A 20-as sorban a gép megnézi a DATA sort, és az első szót C\$-ba, az első számot pedig F-be teszi.

Ez írja ki a C\$-beli szót

A C számláló háromszori ismétlés után leállítja a programot, mivel C\$-ra és F-re csak három DATA egység van

Ez a szubrutin

Valahányszor a program megismétlődik a C\$-ban és az F-ben levő szavak, ill. számok kicserélődnek a következő két DATA értékre

Most a válaszok ellenőrzésére használjuk a szubrutint! A jó válaszokat F tartalmazza, a felelet válasza A-ban van. A szubrutin 100-as és 110-es sora megnézi A és F eltérését. Az eltérést a különbség

abszolút értéke adja (ami mindig pozitív), ezért van az ABS jel. Ha az eltérés 10-nél kisebb, a válasz elfogadható, ha nagyobb, akkor nem.

# Mi mindent lehet csinálni szavakkal?

A legtöbb számítógép a szövegváltozóknak elraktározott szavakat sokféleképpen meg tudja vizsgálni, és mindenféle érdekes műveleteket végez velük. Pontos ellenőrzést a szöveges változók tartalmát, és észreveszi, ha egy különleges szó vagy betű szerepel benne. Betűről betűre követi a begépelte szavakat is. Át tudja rendezni a szavakat vagy betűket, össze tudja fűzni őket, és még sorolhatnánk, hogy mi mindenre képes, de folytassuk inkább a példákkal.

1

A legtöbb számítógépnél  
(de a ZX 81-nél nem)

a LET szó  
elhagyható

```
10 A$ = "BUTA VAGYOK"  
20 B$ = "CSAK A BOLOND HISZI, HOGY"  
30 C$ = B$ + " " + A$  
RUN  
CSAK A BOLOND HISZI, HOGY: BUTA VAGYOK
```

2

```
PRINT LEFT$(B$,4)  
CSAK  
PRINT LEFT$(B$,4) + " " + A$  
CSAK BUTA VAGYOK
```

Két karakteres változó tartalmát így lehet összeadni. Az időzjelek közötti üres helyre azért van szükség, hogy a szavak között meglegyen a szóköz.

3

```
PRINT RIGHT$(A$, 6)  
VAGYOK
```

Ha azt akarjuk mondani a gépnek, hogy vegye a jobb oldali betűket, akkor a RIGHT\$ (RIGHT JOBB) szót gépeljük be a karakter változónevével, és a betűk számát.

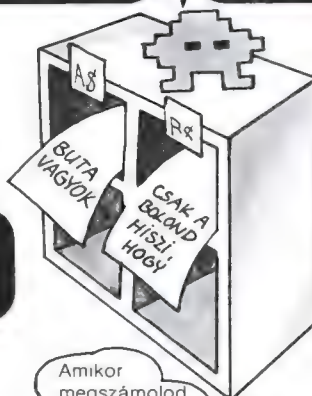
5

```
10 K$ = "BIM BAM!"  
20 PRINT LEN(K$)  
RUN  
8
```

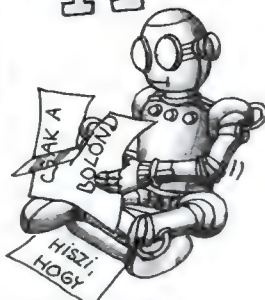
Megtudhatjuk a betűlánc hosszát is — azaz a hozzá tartozó összes betű, szóköz és írásjel számát. Ehhez a LEN szót használjuk, amely a (LENGTH HOSSZÚSÁG) szó rövidítése.



```
IF A$ = "ROBOTHALOM"  
mi a LEFT$(A$, 8)  
mi a RIGHT$(A$, 8)  
mi a MID$(3, 6)
```



Amikor megszámo-  
lod a betűket, be kell  
számítani a szóközöket  
és az írásjeleket is.



A változók részeit is  
összeadhatjuk.

LEFT\$(B\$,4) (LEFT BAL) azt  
jelenti, hogy balról az első  
négy betűjét vegyük B\$-nak.

4

```
PRINT MID$(B$, 8, 6)  
BOLOND
```

Ez arra utasítja  
a számítógépet, hogy középről  
vegyen betűket (MID: MIDDLE:  
középső). Az első számból  
derül ki, hogy honnan kezdje,

**Megjegyzés Sinclair gép  
tulajdonosoknak**

```
PRINT A$(6 TO 11)  
VAGYOK  
PRINT B$(16 TO 20)  
HISZI
```

Ez azt jelenti, hogy  
hogy a hatodiktól  
a tizenkettedik betűig  
kell kiírni.

A Sinclair számítógépek  
nem használják a LEFT\$-t,  
a RIGHT\$-t és a MID\$-t sem.  
A példa szemlélteti, hogy a  
Sinclair gépek ezek helyett  
mit kell mondanunk.



# Titkosírásprogram

Ez a program automatikusan átkódolja a szavakat. Hasonló, csak bonyolultabb programokat használnak a kémiszervezetek a kódok írására és megfejtésére. A programot úgy értjük meg a legkönnyebben, ha egy titkos üzenetet leírunk egy papírra, aztán a program sorait végignézzve végrehajtjuk a számítógép feladatait, és leírjuk az eredményt.

```

5 LET C$=""
7 LET D$=""
10 PRINT "GÉPELJ BE EGY RÖVID SZÖVEGET"
20 INPUT M$
30 PRINT "MOST GÉPELJ BE EGY TITKOS
   SZÁMOT 2 ÉS"; LEN(M$) - 1;"KÖZÖTT"
40 INPUT N
50 LET A$=RIGHTS(M$,N)
60 LET B$=LEFTS(M$, LEN(M$)-N)
70 LET M$=A$+B$
80 FOR I=1 TO LEN(M$) STEP 2
90 LET C$=C$+MID$(M$, I, 1)
100 NEXT I
110 FOR J=2 TO LEN(M$) STEP 2
120 LET D$=D$+MID$(M$, J, 1)
130 NEXT J
140 LET M$=C$+D$
150 PRINT "A KÓDOLT ÜZENET:"
160 PRINT M$
170 END

```

Bevezet két üres szövegváltozót

Ez a szöveg hossza, mínusz 1.

N (a titkos számod) számú betű M\$ jobb végétől számítva

M\$ hossza, mínusz N, számú betű M\$ bal végétől számítva (azaz a maradék betűk)

M\$ helyébe A\$ + B\$ kerül

Az első betűtől kettőssel végighalad a szöveg betűin. Az új M\$ páratlan betűit ez, az I ciklus C\$-ba helyezi M\$ egy betűjét az I pozíciójáról áttesz C\$-ra

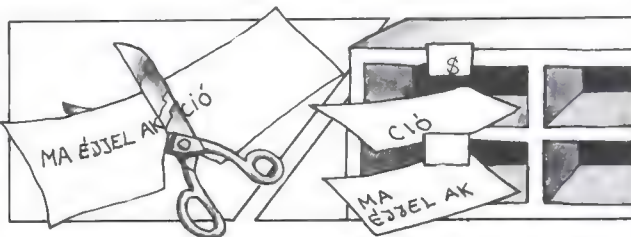
A J ciklus a páros számú betűket helyezi egymás után D\$-ba. A két ciklus hasonlóan működik

Újból visszatesztli a betűket M\$-ba

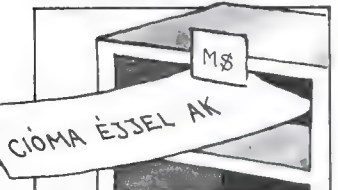
## Hogyan működik?



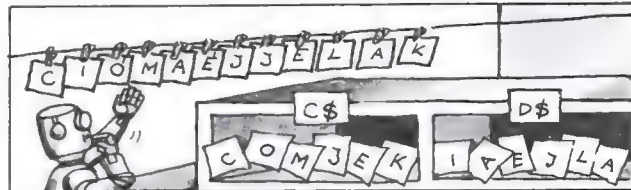
Tegyük fel, hogy az üzenet "Ma éjjel akció", és a titkos számod a 3. Ezeket M\$-ban és N-ben raktározzuk el.



Az 50-es és a 60-as sorban a számítógép a titkos számod alapján kettéosztja az üzenetet. Az 50-es sorban az üzenet három betűjét betesz az A\$-ba. A 60-as sorban a maradék betűket betesz B\$-ba.



A 70-es sorban összeadja A\$-t és B\$-t. Ezzel az üzenet betűit hátulról előre tette.



Az I ciklus minden lépésben áttesz az M\$ páratlan sorszámú betűjét a C\$-ba (C, Ó, A stb.). A J ciklus pedig a páros számú betűket D\$-ba rakja át. Így a C\$ meg a D\$ a titkos üzenet alaposan összekevert betűit írja ki.



# Ábrák és szimbólumok

A számítógép a legkülönbözőbb módokon (pl. szavakkal, számokkal, képekkel, grafikonokkal) képes kifejezni magát. A bonyolultabb összefüggéseket könnyebb megérteni, ha grafikonokkal, képekkel és jelekkel szemléltetted.



Képzeljünk el egy őszibarackfát, amely minden évben a korával arányos mennyiségű gyümölcsöt terem. Ez kifejezhető mondjuk az  $Y = 3X + 2$  egyenlettel ( $Y$  a termés, és  $X$  a kor). Sokkal könnyebb azonban a képletet megérteni, ha egy grafikonnal szemléltetjük.



A géppel nagyon könnyen fel tudjuk rajzolni, hogyan változik  $Y$  az  $X$ -szel arányosan. A grafikon megrajzolásához minden  $X$  értékhez meg kell találni a hozzá tartozó  $Y$ -t. Ez egyszerűen megoldható a  $LET Y = 3 * X + 2$  utasításra épülő programban.



```
5 CLS
10 FOR X=1 TO 14
20 LET Y=3*X+2
30 PLOT (X,Y)
40 NEXT X
50 END
```

Ezen a grafikonon  $X$  maximális értéke 14, és  $Y$  maximális értéke  $3*14+2$ .



Ez a grafikonrajzoló program. A ciklus végigmegy  $X$  összes értékén 1-től 14-ig. A ciklus minden lépésében a 20-as sor  $X$  értékével kiszámítja  $Y$ -t, és a 30-as sor

berajzolja az  $(X,Y)$  pontot a képernyőn. Egy grafikonrajzoló programnál mindig meg kell győződni, hogy  $X$  és  $Y$  maximális értéke ráfér-e a képernyőre!

## A számítógép és a matek

A több lépésből álló számításokban, amilyen a  $3X + 2$  is, a gép először mindig a szorzásokat és az osztásokat végzi el, és csak azután az összeadásokat és a kivonásokat. Tehát a gép ugyanazt a megoldást adja erre a két számításra:

```
PRINT 4*6+8      PRINT 8+4*6
32                32
```

Ha a műveleteket más sorrendben akarjuk a géppel elvégeztetni, zárójelet kell használnunk, pl.

```
PRINT(8+4)*6
72
```

Ezúttal előbb az összeadásra kerül sor  $(8+4)$  és az összeget szorozzuk 6-tal.

## Programfejlesztő

GONDOLJ KI EGY SZÁMOT  
VEDD A DUPLÁJÁT, ADJ HOZZÁ 4-ET  
OSZD EL 2-VEL, ADJ HOZZÁ 7-ET  
SZOROZD MEG 8-CAL, VONJ KI 12-T  
OSZD EL 4-GYEL ÉS VÉGY EL 11-ET  
MONDD MEG AZ EREDMÉNYT  
A SZÁM, AMIT ELŐSZÖR  
KIGONDOLTÁL

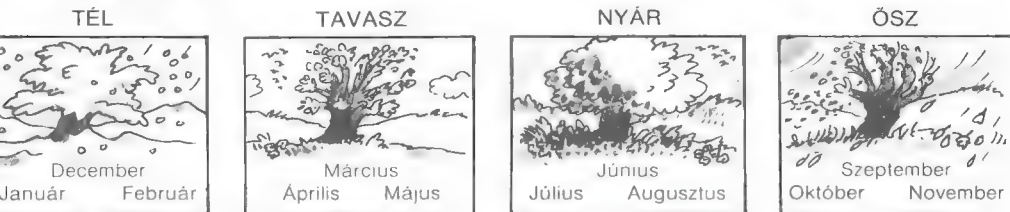
Tudnál írni egy programot erre a jól ismert fejszámolótrükkre? (A kigondolt számot úgy kapod meg, hogy az eredményből 4-et kivonsz és azután 2-vel osztod.)

# Születésnapprogram

Ez a program másképpen szemlélteti az eredményeket a képernyőn. "\*" jeleket használ a különböző évszakokban született emberek számának bemutatására. Ilyen fajta programmal lehet mindenféle mérési adatot összehasonlítani, pl. madarak észlelési adatait, vagy futballcsapatok eredményeit. Mielőtt hozzáfogunk ehhez a már egész komoly programhoz, jó, ha előbb programtervet készítünk!

## Programterv

**A cél:** A télen, tavasszal, nyáron és ősszel született emberek számának összehasonlítása.



1. Add meg a gépnek egy 20-fős vizsgálat adatait (vagyis az évszakokat, amikor az emberek születtek)!
2. Tárold az adatokat a gépben!
3. Ird ki az adatokat a képernyőre!

## A program

```
5 LET A = 0
6 LET B = 0
7 LET C = 0
8 LET D = 0
```

A változók készen állnak a számlálásra.

```
10 FOR I = 1 TO 20
20 PRINT I; "SZÁMÚ SZEMÉLY SZÜLETETT"
30 PRINT "TÉLEN, TAVASSZAL, NYÁRON VAGY ŐSZELEN"
40 PRINT "TÉ-TA-NY-Ő"
50 INPUT B$
60 IF B$ = "TE" THEN LET A = A + 1
70 IF B$ = "TA" THEN LET B = B + 1
80 IF B$ = "NY" THEN LET C = C + 1
90 IF B$ = "Ő" THEN LET D = D + 1
```

A ciklus arra való, hogy a gép végigkérdezze a kísérleti alanyok adatait.

A 60-asról a 100-as sorig a B\$-ban megőrzött választ vizsgálják, és a megfelelő évszak számlálójához hozzáadnak egyet.

```
100 NEXT I
110 PRINT "TÉLI ÖSSZLETSZÁM"
115 LET N = A
120 GOSUB 200
130 PRINT "TAVASZI ÖSSZLETSZÁM"
135 LET N = B
140 GOSUB 200
150 PRINT "NYÁRI ÖSSZLETSZÁM"
155 LET N = C
160 GOSUB 200
170 PRINT "ŐSZI ÖSSZLETSZÁM"
175 LET N = D
180 GOSUB 200
190 STOP
```

Visszaküldi egy újabb kérdésre.

A szubrutin annyi csillagot ír ki, amennyi az egyes változók értéke

Az összletszám mindig az N változóban van, így mindegyik évszakra ugyanazt a szubrutint alkalmazhatjuk

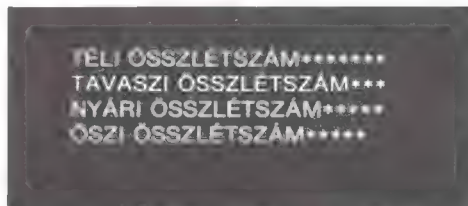
A csillagok ugyanebbe a sorba kerülnek

```
200 REM: SZUBRUTIN CSILLAGOK
    KINYOMTATÁSÁRA
210 IF N = 0 THEN GOTO 250
220 FOR I = 1 TO N
230 PRINT "*"
240 NEXT I
250 PRINT
260 RETURN
```

Ha senki sem született egy adott évszakban, akkor nem kell a szubrutinra lépni

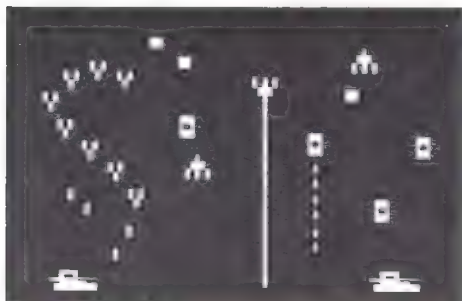
A főprogram N-be helyettesíti A, B, C, D értékét  
A ciklus tehát minden esetben N csillagot rajzol ki.

## Próba futtatás

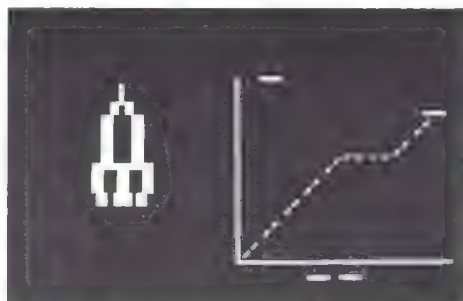


# Mozgó képek

Ezen a két oldalon bemutatjuk, hogy a **PLOT** és **UNPLOT** utasításokkal hogyan lehet a képernyőre mozgó képeket rajzolni. Mozgó képek helyett animációt is szoktak mondani, mindenesetre igen jól hasznosíthatók játékokban, vagy különböző elméleti mozgáspályák (egy elhajított kő vagy a Hold Föld körüli útja) szemléltetésére.



A képet egy videójátékból vettük. Egyes kis számítógépeket úgy programoznak, hogy csak játszani lehet velük. Ezek a programok nem BASIC-ben, hanem a gép saját kódjában készülnek.



Egy BASIC-ben programozott, általános célú mikroszámítógép lassúbb és egyszerűbb képeket csinál. A képernyő utasításait nem tudja elég gyorsan és egyszerre kezelni, így a képek nem mozognak elég gyorsan.

## PLOT/UNPLOT program

1

```
10 LET X=1
20 LET Y=1
30 PLOT (X,Y)
40 UNPLOT (X,Y)
50 LET X=X+1
60 LET Y=Y+1
70 GOTO 30
```

Bizonyos gépeken át kell térni a grafikus üzemmódra!



2

Igy mozog a fénypont! Most még világít, de a következő pillanatban már kialszik.



Ez a rövid program végigmozgat egy fénypontot a képernyőn. Ne felejtse el, hogy a **PLOT** és az **UNPLOT** utasítás gépenként változó!

Amikor a pont eléri a képernyő szélét, a program hibaüzenettel leáll, hiszen **X** és **Y** értékei túllépi a képernyő tartományát.

3

LET X=X+1

LET Y=Y+1



LET Y=Y-1

LET X=X-1

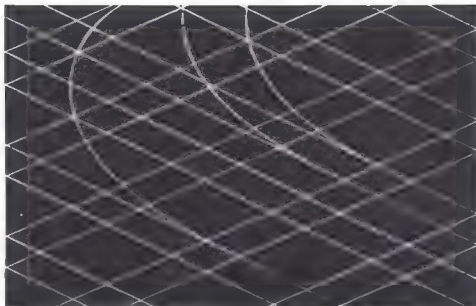
A fenti programot felhasználhatjuk pl. egy labda mozgásának utánzására. Közben persze be kell tartanunk néhány programozási szabályt, hogy a labda le ne "guruljon" a képernyőről.

Amikor a labda eléri a képernyő tetejét, csökkentjük az **Y**-t, és ha az aljáig jut, növeljük. Ugyanez vonatkozik az **X**-re is, bal oldalt növelnünk, jobb oldalt csökkentenünk kell.



# Vonalhúzó program

Most egy olyan program következik, amelyik keresztül-kasul vonalakat húz a képernyőn. Amikor a vonal eléri a képernyő szélét, visszafordul és egy másik irányban halad tovább. Itt nem használjuk az **UNPLOT**-ot, tehát a mozgó pontok nyoma ottmarad a képernyőn. A kép egy programfutás eredményét mutatja. A program **100**-as sora **10** ezer pont kirajzolását állítja be. Ha akarod, kicserélheted a **10** ezret kisebb számra, vagy **BREAK**-kel is megszakíthatod a rajzolást.



10 REM: HA SZÜKSÉGES, ITT IKTASS BE EGY GRAFIKUS MÓDOT

20 PRINT "HÁNY KÉPPONT LEGYEN"  
A VÍZSZINTESEN";

30 INPUT V

40 PRINT "ÉS FÖLFELÉ"

50 INPUT F

55 CLS

60 LET X = V/2

70 LET Y = F/2

80 LET S = (INT(RND(1)\*10 + 1) - 5)/50

90 LET T = 1

100 FOR I = 1 TO 10000

110 LET X = X + S

120 LET Y = Y + T

130 IF X < 5 THEN LET S = -S

140 IF X > V - 5 THEN LET S = -S

150 IF Y < 5 THEN LET T = -T

160 IF Y > F - 5 THEN LET T = -T

170 GOSUB 300

180 NEXT I

200 STOP

300 REM: PLOT SOR

310 PLOT(X,Y)

320 RETURN

A 20-as és 30-as sor a képernyő szélességét és magasságát kérdezi meg.

X és Y kezdőértéke a képernyő közepe.

Az S és T-beli összegeket X-hez és Y-hoz adva hozzuk létre a vonal mozgását

Innentől a 190-es sorig tartó ciklus 10000-szer ismétlődik. X és Y értéke minden lépésben egy kicsit változik

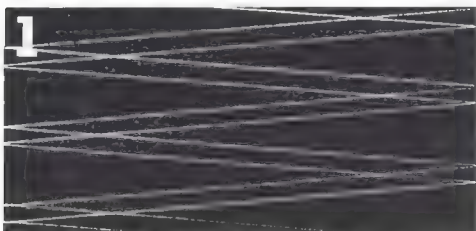
Ez egy nagyon kicsi számot ad, amit X-hez adunk. A nagyon kicsi szám minden lépésben más és más

Ezek a sorok vizsgálják a széleket. S és T előjelét megváltoztatják, amikor X és Y 5 képpontnyira van a szélektől.

Most a gép rátér a rajzolószubrutinra.

Kirajzolja az X és Y pillanatnyi értékével a képpontot

## Kísérletek



A 110-es sor minden alkalommal egy nagyon kicsi számot ad hozzá X-hez, és ettől megy a vonal keresztül-kasul a képernyőn. Ha pl. kitörölöd ezt a sort, a vonalak párhuzamosakká válnak.

Próbáld megváltoztatni a számokat a 80-as és a 90-es sorban mondjuk 5-tel vagy 10-zel (vagy még nagyobb számmal, ha finom felbontású grafikád van)! Ettől a vonalak szaggatottá válnak.

# Egy furcsa költemény

A következő pár oldalon egy versíró programot mutatunk be, amit nemcsak igazi számítógéppel, hanem papíron is le lehet futtatni. Egy hosszú csík tartalmazza a programsorokat, egy pörgettyűre van bízva a véletlen számok előállítás, és táblázatok tartalmazzák a vers alkotóelemeit.

## A "papír számítógép" programja

## Adatsorok

VOLT EGY EMBER .KI  
AKI A BARÁTJA  
EGYSZER PEDIG, MIKOR LESZÁLT AZ ESTE  
MINDENT ELFELEJTETT

## Szótáblázat

VEZSPREMBŐL	BÉKÉSRŐL	CEGLÉDRŐL	VECSÉSRŐL
ÁLMODOTT	BESZÉLT	MESÉLT	OLVASOTT
BUTA	TUNYA	SUTA	MULYA
FEJÉRŐL	MENYÉRŐL	SZEMÉRŐL	NEJÉRŐL
VÁGYÓDOTT	ÁGYAZOTT	TRÁGYÁZOTT	KÁRTYÁZOTT
ÉS A TEJET FEJTE	ÉS A HOLDAT LE	ÉS A KÉSÉT FENTE	ÉS ALUDT A BESTE
AZ EGÉSZBŐL	AZ EBÉDRŐL	A NEVÉRŐL	A REMÉNYRŐL

VOLT EGY EMBER VECSESRŐL,  
AKI ÁLMODOTT A BARÁTJA SUTA MENYÉRŐL,  
EGYSZER PEDIG, MIKOR LESZÁLT AZ ESTE,  
TRÁGYÁZOTT ÉS A HOLDAT LESTE  
MINDENT ELFELEJTETT A REMÉNYRŐL.

- 1 A=0 és B=0
- 2 A-hoz adj hozzá 1-et!
- 3 Ha A=6, menj a 10-es sorra!
- 4 Írd ki az A-adik adatsort!
- 5 Adj B-hez 1-et!
- 6 A pörgettyűvel válaszd ki N-et!
- 7 Írd ki a B-edik sor  
N-edik oszlopában álló szót!
- 8 Ha B=3 vagy 5, menj  
az 5-ös sorra!
- 9 Menj a 2-es sorra!
- 10 Állj

Ez a papír számítógép programja. Majdnem olyan, mint a BASIC, de azért így még nem működne egy valódi számítógépen. A szavak és kifejezések egy

külön papíron sorakoznak, és a program dönti el, mikor melyikre kerüljön sor. A pörgettyű 1 és 4 közötti véletlen számot ad.

## I Lefordítjuk a programot BASIC-re

```
10 LET A=0
20 LET B=0
30 LET A=A+1
40 IF A=6 THEN STOP
50 Írd ki az A-adik adatsort
60 LET B=B+1
70 LET N=INT(RND(1)*4+1)
80 Írd ki a B-edik sor
   N-edik oszlopában levő szót
90 IF B=3 THEN GOTO 60
100 IF B=5 THEN GOTO 60
110 GOTO 30
120 END
```

Ez még nem fog  
működni a gépen.



Ezek a sorok beállítják a kezdeti értéket.

A 30-as és a 40-es sor számolja a már kiválasztott adatsorokat.

Az 50-es és a 80-as sor még nincs lefordítva BASIC-re.

Itt számoljuk az adattáblázat szavait.

1 és 4 közötti véletlen számot ad.

A 90-es és a 100-as sor visszaküldi a programot egy újabb adatsorért.

A program legnagyobb részét könnyű volt BASIC-re lefordítani, de az 50-es és 80-as sor már nehezebb lesz. Meg kell oldanunk,

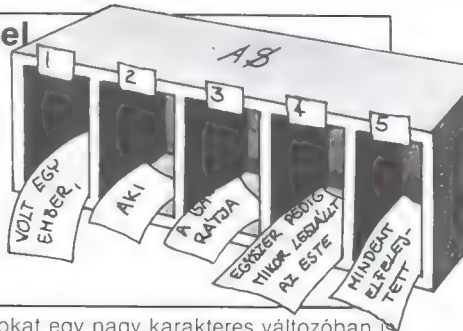
hogy a gép tárolja és ki is tudja olvasni a verssorokat és az adatszavakat.

## 2 Adatok tárolása számítógéppel

50 READ A\$

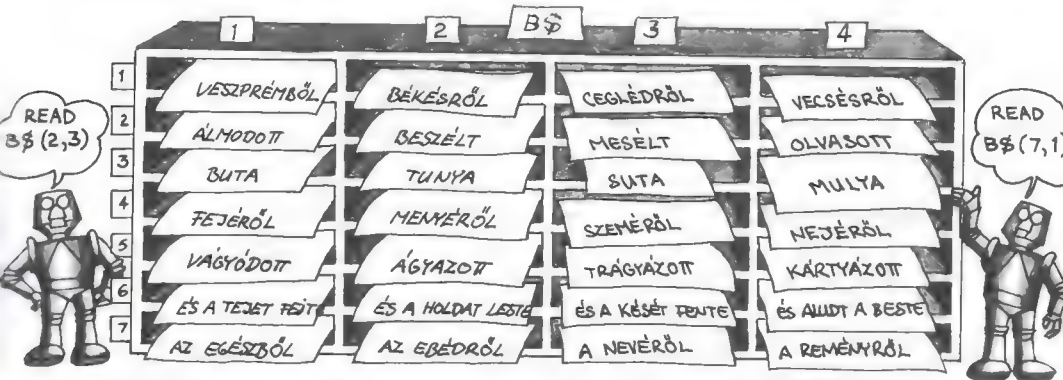
180 DATA "VOLT EGY EMBER," "AKI," "BARÁTJA"

190 DATA "EGYSZER PEDIG MIKOR LESZÁLLT ,  
AZ ESTE," "MINDENT ELFELEJTETT"



Adatokat a **DATA** és **READ** utasítások segítségével lehet megadni. Minden alkalommal, amikor a **READ** utasításon áthalad a program, egy újabb adatot olvas be a **DATA** sorból a változóba.

Az adatokat egy nagy karakteres változóban is tárolni lehet. Azokat a változókat, amelyekben egynél több adatot is elraktározhatunk, tömbnek hívjuk. Mindegyik elemet egy sorszámmal jelölhetjük meg.



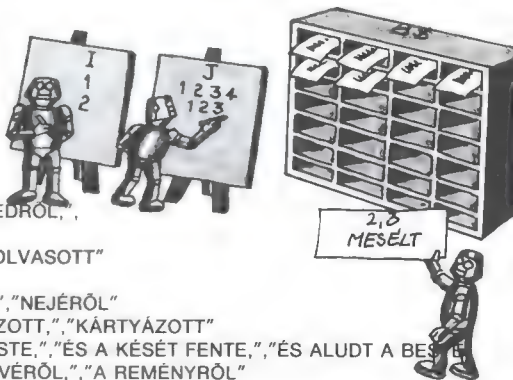
A változó több sorban is tarolhat szavakat, így egy egész táblázatot is. Ilyenkor minden szót a sorával és az oszlopával határozzuk meg. Például

a B\$(4,2) tartalma: "MENYÉRŐL", vagy a B\$(6,3)-é: "ÉS A KÉSÉT FENTE". Számokat is tarolhatunk így, csak akkor a változó nevében nem kell a dollárjel.

### Az adatokat beolvassuk a változóba

```
10 FOR I = 1 TO 7 ] I a sorszám.
20 FOR J = 1 TO 4 ] J az oszlopszám.
30 READ B$(I, J)
40 NEXT J
50 NEXT I
```

```
60 DATA "VESZPRÉMBŐL," "BÉKÉSŐRŐL," "CEGLÉDRŐL," "VECSÉSRŐL"
70 DATA "ÁLMODOTT," "BESZÉLT," "MESELT," "OLVASOTT"
80 DATA "BUTA," "TUNYA," "SUTA," "MULYA"
90 DATA "FEJÉRŐL," "MENYÉRŐL," "SZEMÉRŐL," "NEJÉRŐL"
100 DATA "VÁGYÓDOTT," "ÁGYAZOTT," "TRÁGYÁZOTT," "KÁRTYÁZOTT"
110 DATA "ÉS A TEJET FEJT," "ÉS A HOLDAT LESTE," "ÉS A KÉSÉT FENTE," "ÉS ALUDT A BESTE"
120 DATA "AZ EGÉSZBŐL," "AZ EBÉDRŐL," "A NEVÉRŐL," "A REMÉNYRŐL"
```



Ugy lehet az adatokat sorra beolvasni, hogy a **READ** utáni változóban a zárójelen belüli számértéket változtatjuk. Ezt egy ciklussal tudjuk megoldani. Az I ciklus a sorokon, a J ciklus az

oszlopokon halad végig. Egy adott I-hez, ami 4-féle lehet, a J végigfut az egész oszlopon, vagyis 7 értéken. Így minden adat sorra kerül.



## 5 Hogyan biztosítsunk helyet a változónak?

```

5 DIM K$(5) ]
10 FOR I = 1 TO 5 ]
20 READ K$(1) ]
30 NEXT I ]
40 STOP ]

```

Ez adja meg a változó méretét. Itt 5 elem kerülhet a sorba

A ciklus sorban egymás után beteszi az adatokat a K\$ változóba

60 DATA TŰZ, VÍZ, JÉG, ESŐ

A program elején meg kell adni a gépnek, hogy mekkora helyet foglaljon le a változó számára. Ezt a **DIM** (DIMenzió: kiterjedés) utasítással oldhatod meg, mögé írva a változó nevét. Példánkban: **DIMK\$(5)**

## 6 Az adatok kinyomtatása

```

200 LET A=0
210 LET B=0
220 LET A=A+1 ]
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1 ]
260 LET N=INT(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250 ]
290 IF B=5 THEN GOTO 250 ]
300 GOTO 220 ]

```

A gépnek ezekre a sorokra van szüksége, hogy kinyomtassa megfelelő elrendezésben a rögzített és a változó vesszorokat. Minden rögzített vesszor után, amit A-val számolunk, következnek

Az A számolja, hogy hányszor ismétlődik a program.

A B számolja a sorokat a szótáblázatban.

A 280-as, 290-es sor hatására egymás után két szót választunk a szótáblázatból.

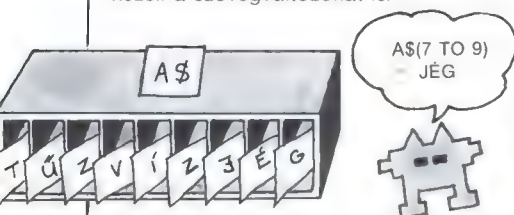
Ez küldi vissza a gépet a következő rögzített vesszor kiírására.

a szótáblázatból véletlenszerűen választott szavak, B határozza meg, hogy hányadik sorból, N pedig, hogy melyik oszlopból választunk.



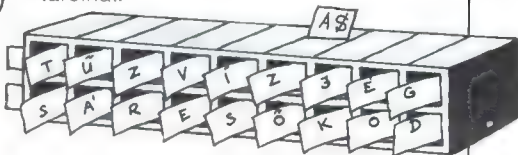
## A ZX 81-en mindez másképp van

A ZX 81 utasításai között nincs se **DATA**, se **READ**, egész másképp kezeli a szövegváltozókat is.



Egy adott szót úgy választhatunk ki a hosszú szövegváltozóból, hogy az első és az utolsó betűjének a sorszámát adjuk meg, ahogy már a 32-ik oldalon is láthattad.

Karakteres tömböknél pontosan meg kell adni a gépnek, hogy milyen hosszúak lehetnek a sorok. Például **DIM A\$(2,4)** két 4-karakteres sort tárolhat.



A program elején a ZX 81-nek is meg kell mondani, hogy mekkora területre lesz szükséged egy tömbhöz, vagyis az oszlopok és sorok számát. A formája ugyanaz, mint az előbb, például **DIMAS\$(2,9)**. Most mindegyik sorban ugyanolyan hosszú szavak kellene.

# A teljes versprogram

Az eddigi részletek alapján most már összeállíthatjuk a teljes programot. Az első rész (10-től 190-ig) megadja a gépnek az adatokat, a második rész (200-tól 310-ig) kinyomtatja a verssorokat. Ha többször futtatod a programot, mindig új vers íródik ki, mert a véletlen N szám miatt más és más szavak kombinálódnak.

```

10 DIM A$(5)
20 DIM B$(7,4)
30 FOR I=1 TO 7
40 FOR J=1 TO 4
50 READ B$(I,J)
60 NEXT J
70 NEXT I
80 DATA "VESZPRÉMBÖL","BÉKÉSRÖL","CEGLÉDRÖL","VECSÉSÖRÖL,"
90 DATA "ÁLMODOTT","BESZÉLT","MESELT","OLVASOTT,"
100 DATA "BUTA","TUNYA","SUTA","MULYA,"
110 DATA "FEJÉRÖL","MENYÉRÖL","SZEMÉRÖL","NEJÉRÖL"
120 DATA "VÁGYÓDOTT","ÁGYAZOTT","TRÁGYAZOTT","KÁRTYÁZOTT,"
130 DATA "ÉS A TEJET FEJTE","ÉS A HOLDAT LESTE","ÉS A KÉSÉT FENTE","ÉS ALUDT
    A BESTE,"
140 DATA "AZ EGÉSZBÖL","AZ EBÉDRÖL","A NEVÉRÖL","A REMÉNYRÖL,"
150 FOR I=1 TO 5
160 READ A$(I)
170 NEXT I
180 DATA "VOLT EGY EMBER","AKI","A BARÁTJA"
190 DATA "EGYSZER PEDIG, MIKOR LESZÁLLT AZ ESTE","MINDENT ELFELEJTETT"
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END

```

A 10-es és 20-as sor utasítása: **A\$** számára 5 sort, **B\$** számára pedig 7 sort és 4 oszlopot foglal le.

Ezzel a ciklussal töltjük be a **B\$**-ba a **DATA** sorokból a szótáblázatot.

Ez a szótáblázat kerül **B\$**-ba.

Ez a ciklus tölti fel a rögzített verssorokkal **A\$**-t.

A 180–190-es sorok szavai kerülnek az **A\$**-ba.

Itt írjuk ki az **A**-adik **A\$**-beli szót.

Itt pedig a kiválasztott **B** sor **N** oszlopbeli szót írjuk ki

A program a 230-adik soron áll meg, amikor **A=6**, a 310. sorra tehát sose jut el. Mégis, néhány gépen kotelező **END**-del befejezné a programot

## Példafuttatások

VOLT EGY EMBER CEGLÉDRÖL, AKI MESELT A BARÁTJA BUTA SZEMÉRÖL, EGYSZER PEDIG, MIKOR LESZÁLLT AZ ESTE, ÁGYAZOTT, ÉS A KÉSÉT FENTE, MINDENT ELFELEJTETT AZ EBÉDRÖL	VOLT EGY EMBER VESZPRÉMBÖL, AKI BESZÉLT A BARÁTJA MULYA NEJÉRÖL, EGYSZER PEDIG, MIKOR LESZÁLLT AZ ESTE, VÁGYÓDOTT, ÉS ALUDT A BESTE, MINDENT ELFELEJTETT AZ EGÉSZBÖL
--	---

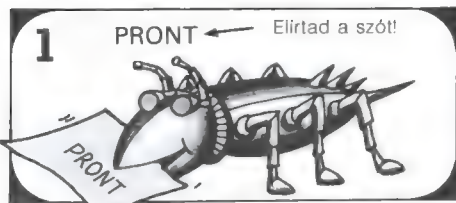
Ez csak két vers a lehetséges 16 384 közül. Ha sokszor kipróbálsz a programot, mindig hasonló verseket kapsz, de egy kicsit mindig mást. Ezzel azt is követheted, hogy hogyan választ

véletlen számokat a gép. Megfigyelheted, hogy némelyik számítógép a bekapcsolás után mindig ugyanazokkal a véletlen számokkal kezd.

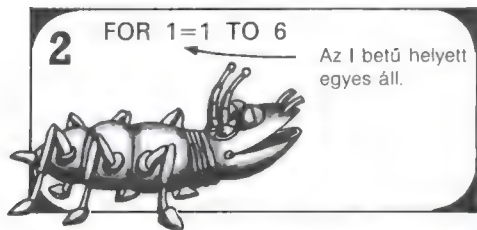
# Programozási ötletek

Ezen a két oldalon néhány tippel, ötlettel segítünk, hogy egy kicsit megkönnyítsük a programírást, és a leggyakrabban előforduló hibák felfedezését. Ha nem működik a programod, fúsd át ezt a listát, hátha éppen az itt felsorolt hibák egyike a ludas!

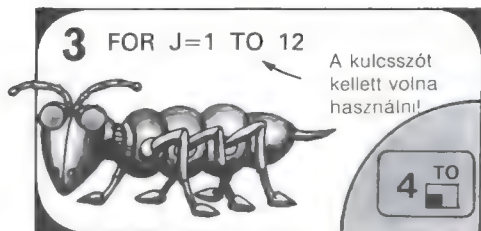
## Hibafajták



A BASIC szavakban még véletlenül sem lehet pontatlanság, betűelírás. A rosszul betűzött utasításokat a gép nem érti meg.



Ellenőrizd a 0-kat és az O-kat, valamint az 1-eseket és az I-ket, nehogy összekevered őket a gépelésnél!



Ha Sinclair géped van, a kulcsszavakhoz tartozó billentyűt kell lenyomni, nem betűnként írni a szavakat!

## Programírás

Programírás közben gondold arra, hogy a gép lényegében háromféle dologra képes: az alapvető utasítások végrehajtására, a megadott lépések ismétlésére és döntésekre.



```
LET A=3  
LET N=N+1  
PRINT A/T  
PLOT (X,Y)
```



```
FOR J=1 TO 6  
20 LET A=1  
30 IF A<10 THEN  
GOTO 100
```



```
IF X=Y THEN STOP  
IF K$="HELLO"  
THEN PRINT A
```

Ebben a kis könyvben az összes alapvető BASIC utasítást összefoglaltuk, így most már valamelyest látod, hogy milyen feladatokat tud végrehajtani egy számítógép. Amikor programot írsz, először a főbb lépéseket dolgozd ki, azután bontsd le BASIC utasításokra!

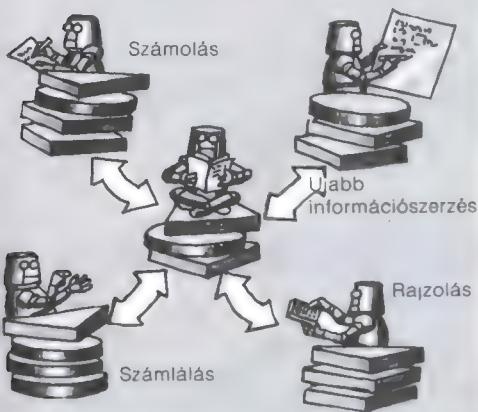


Győződj meg róla, hogy nem felejtettél-e ki egy idézőjelet vagy egy vesszőt például a DATA egységek között. A bonyolult, hosszú sorokat, amelyekben sok jel fordul elő, nagyon gondosan ellenőrizd!

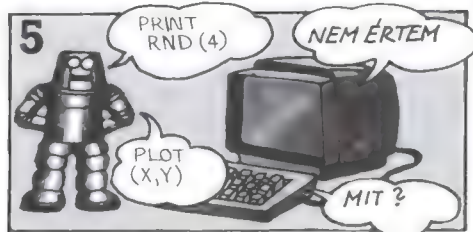


# Hibaüzenetek

Egy programot sokféleképpen lehet megírni, lehet röviden és áttekinthetően, és lehet hosszadalmasabban is. Ha egy hosszabb programot írsz, legjobb, ha a feladatot részfeladatokra bontod, és minden ilyen részfeladathoz írsz egy szubrutint, ami azt elvégzi. A program központi magja egyszerű utasításokból, döntésekből és ismétlésekből állhat, amelyek megszabják, mikor és hányszor hajtsa végre a gép a szubrutinokat.



Ha az egyes részek között megszakítod a programot, sokkal könnyebben megtalálod a hibákat. Így a részleteket önmagukban ellenőrizheted, és nem kell végigbogarásznod az egész programot. Egyébként ne felejsd el eléjük egy REM sort beilleszteni, így később is tudni fogod, hogy miről volt szó!



Ellenőrizd, hogy a helyes formában adtad-e meg az RND, PLOT és CLS utasításokat! Ellenőrizd azt is, hogy szükség esetén állítítottad-e a gépet grafikus üzemmódra!

Minden gép valamilyen üzenetet ír ki, ha hibát talál a programban. Az üzenetek jelentését a gépkönyvedben megtalálod, de a leggyakoribbakat itt is összefoglaljuk.

Out of data



Out of data (kevés az adat) azt jelenti, hogy már nem tud a gép több adatot kiolvasni, elfogytak az adatok, vagyis a data-egységek. Ennek például az lehet az oka, hogy kimaradt egy vessző két adat között, vagy egyszerűen elszámoltad magad.

No such line (nincs ilyen sor) közli a gép, ha GOTO vagy GOSUB utasítással egy nem létező sorra küldted. Lehet, hogy véletlenül kitörölted azt a sort? Vagy csak elírtad a számot?

No such line



No such variable



No such variable (nincs ilyen változó) üzenetet a Sinclair gépek írnak ki, ha nem adsz kezdőértéket a változóknak. PI. LET C=0 vagy LET CS= ""

FOR without NEXT (FOR NEXT nélkül) világosan megmondja, hogy valamilyik ciklust elfelejtetted NEXT-tel lezárni. Az is lehet, hogy rossz változónevet írtál, esetleg I helyett 1-et

FOR without NEXT



## Búcsúzóul

Szó ami szó, olykor nagyon nehéz megtalálni a hibát, és a program bizony addig nem jól fut, amíg a hiba ki nincs javítva. Ilyenkor a gyanús sorokat inkább írd be újra, írsd ki lépten nyomon a változók pillanatnyi értékét, de semmi esetre se add fel!

# A fejtörők megoldásai

## 15. oldal

### Név- és üzenetprogram

```
10 PRINT "HOGY HÍVNAK"  
20 INPUT N$  
30 PRINT "SZIA"  
40 PRINT N$  
50 PRINT "HOGY VAGY"
```

## 17. oldal

### 1. Összeadás

```
10 LET A = 9  
20 LET B = 7  
30 PRINT A*B  
40 PRINT A/B  
50 LET A = A + 1  
60 LET B = B + 3  
70 PRINT A*B, A/B  
80 END
```

Vessző, hogy üres hely maradjon.

## 2. Szorzás

```
30 PRINT A;"SZOR";B;"AZ";A*B  
40 PRINT A;"OSZTVA";B;"-VEL AZ";A/B
```

## 3. Név és üzenet

```
10 PRINT "HOGY HÍVNAK"  
20 INPUT N$  
30 PRINT "SZIA"; N$;"HOGY VAGY"
```

## 18. oldal

### Helyes — helytelen?

```
10 PRINT "MENNYI 7-SZER 7"  
20 INPUT A  
30 IF A = 49 THEN PRINT "REND BEN"  
40 IF A < > 49 THEN PRINT "NEM"; 7*7
```

Az idézőjel után egy pontosvessző kell.

## 19. oldal

### Az életkor-kitaláló program könnyítése

Cseréld ki a 30-as sort, és írd be egy új sort 35-ös számmal:

```
30 IF G < 14 THEN PRINT "ÖREGEBB"  
35 IF G > 14 THEN PRINT "FIATALABB"
```

## 23. oldal

### Rajzszámláló

```
5 LET C = 0  
45 LET C = C + 1  
50 IF C < 6 THEN GOTO 10
```

## Monogramrajzolás

Egy L betű kirajzolását adjuk meg.

```
10 LET X = 15  
20 LET Y = 30  
30 PLOT (X,Y)  
40 LET Y = Y - 1  
50 IF Y > 5 THEN GOTO 30  
60 LET X = X + 1  
70 PLOT (X,Y)  
80 IF X < 45 THEN GOTO 60  
90 END
```

## 24. oldal

### Véletlen számok 10 és 20 között

A képlet:  $\text{INT}(\text{RND}(1) * 11 + 9)$ .

Olyan gépeknél, amelyeknél az RND után csak egy zárójeles számot kell megadni,  $\text{RND}(11) + 9$ . A 10 és 20 között tizenegy lehetséges szám van, így 1 és 11 között véletlen számokat kell előállítani, és 9-et hozzáadni.

## 25. oldal

### Találatszámoló

Ezeket a sorokat illeszd be az úrtámadásprogramba:

```
15 LET S = 0  
75 IF X = A*B THEN LET S = S + 1  
95 PRINT "6-BÓL"; S; "ELLENSÉGGEL  
VÉGEZTÉL"
```

## 27. oldal

### 1. Nyolcas szorzótábla

```
10 PRINT "NYOLCAS SZORZÓTÁBLA"  
20 FOR J = 1 TO 12  
30 PRINT J; " * 8 = " J * 8  
40 NEXT J
```

## 27. oldal

### 2. Általános szorzótábla

```

10 INPUT "ÍRJ BE EGY SZÁMOT";N
20 PRINT "ITT VAN";N;"SZORZÓTÁBLÁJA"
30 FOR I=1 TO 12
40 PRINT I;"-SZER";n;"AZ" I*N
50 NEXT I
60 INPUT "ÍRJA KÍ EGY MÁSIK
   SZORZÓTÁBLÁT (I VAGY N)";M$
70 IF M$="I" THEN GOTO 10

```

A ZX 81 esetében külön PRINT és INPUT sorok kellene!

## 32. oldal

### Betűjáték

```

LEFT$(A$,8) "BOTHALOM"
RIGHT$(A$,10) "ROBOTHAL"
MID$(A$,5,8) "BOTHAL"

```

## 34. oldal

### Fejlesztő program

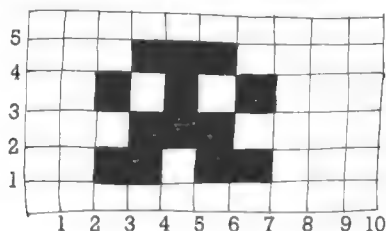
```

10 PRINT "GONDOLJ EGY SZÁMRA"
20 PRINT "DUPLÁZD MEG,ADJ HOZZÁ 4-ET"
30 PRINT "OSZD EL 2-VEL,ADJ HOZZÁ 7-ET"
40 PRINT "SZOROZD MEG 8-CAL,VONJ KI
   12-T"
50 PRINT "OSZD EL 4-GYEL ÉS VONJ LE
   11-ET"
60 PRINT "MONDD MEG AZ EREDMÉNYT"
70 INPUT N
80 PRINT "A SZÁM,AMIRE ELŐSZÖR
   GONDOLTÁL:";(N-4)/2

```

! Azért van szükség a zárójelekre, hogy a gép a megfelelő sorrendben végezze el a műveletet.

## Ürtámadók ismétlőprogramja (A 29. oldalhoz)

**1**

Rajzold meg az ürtámadó mintáját egy kockás papírra!

**2**

4,5; 5,5; 6,5  
 3,4; 5,4; 7,4  
 4,3; 5,3; 6,3  
 3,2; 4,2; 6,2; 7,2

Azután számítsd ki az ürtámadó valamennyi koordinátáját!

**3**

```

5 CLS
50 INPUT "HÁNY PONT VAN A KÉPERNYŐN KERESZTBEN";W
60 INPUT "HÁNY VAN FÖLFELE";V

```

```

60 CLS
70 FOR I=0 TO V-1 STEP V/6
80 FOR J=0 TO W-1 STEP W/6

```

```

130 NEXT J
140 NEXT I
150 END

```

Változtasd, növeled a 6-ot, így növeled a támadóminta ismétlődését a képernyőn! (Ha hibauzenetet kaptál, akkor túl nagy számot választottál!)

Ide kerülnek a kirajzolósorok, pl.:

```

90 PLOT (J+3,I+2)
92 PLOT (J+4,I+2)

```

Az ürtámadó 13 kis kockáját külön kell kirajzoltatnod!

A mintaismétlő programtól a véletlen mintát előállító sorokat (10-estől 40-ig és 90-estől 140-esig) cseréld ki az ürtámadót kirajzoló sorokra, a 80-as és 140-es sor közé!



# BASIC alapszavak

A könyvben előforduló BASIC alapszavak listáját rövid magyarázatokkal egészítettük ki. Csillaggal jelöltük közülük azokat, amelyeket nem minden gép ért (ilyen pl. a CLS). Ha megnézed a gépkönyvet, kiderül, hogy a te géped mit használ ezek helyett.

**\*BREAK** A legtöbb gépnél a program leállítására való. Vigyázat! Egyes gépeknél ilyenkor a programot ki is törli a memóriából, helyette az **ESCAPE** vagy valamilyen más alapszó használható.

**\*CLS** Letörli a képernyőt.

**\*DATA** Adatok (azaz betűk vagy számok) listáját vezeti be, amelyeket azután **READ** utasítással változóba lehet beolvasni.

**DIM** Egy változó számára bizonyos nagyságú memóriaterületet foglal le. Pl. **DIM A\$(5,4)** azt jelzi, hogy az **A\$** változó 5 sort és 4 oszlopot igényel.

**\*EDIT** Egy programsor módosítására való (ha nem akarsz az egész sort újra beadépelni).

**\*END** A program végét jelzi. Egyes gépeknél kötelező, míg pl. a Sinclair gépek nem is ismerik.

**FOR...NEXT** Azt jelzik a számítógépnek, hogy a köztük álló utasításokat megadott számú ciklusban ismételtelen végre kell hajtani.

**GOSUB** Hatására a számítógép megszakítja a főprogram végrehajtását, és egy részfeladat megoldására a megjelölt sorszámu szubrutinra ugrik.

**GOTO** Egy másik programsorra való ugrást ír elő.

**IF...THEN** Adatokat (számokat, szavakat, változókat) hasonlít össze, és az eredménytől függően hajt vagy nem hajt végre egy megadott utasítást.

**INPUT** Amikor a program ideér, megáll, és adatokat kér.

**INT** Tizedes törtet egész számmá alakít úgy, hogy a tizedesponttól jobbra álló jegyeket egyszerűen elhagyja.

**\*LEFT\$** Egy karakterlánc elejéről vesz adott számú karaktert. Pl. **LEFT\$(A\$,4)** **A\$** első négy karakterét jelenti.

**LEN** Egy karakterlánc hosszát (vagyis karaktereinek a számát) adja meg.

## Számítógépes szavak

**Bug** (poloska) Valamilyen hiba a programban.

**Blokkdiagram** A program főbb műveleteit feltüntető vázlat. Gyakran használják a program megtervezésére is.

**CPU** A számítógép központi feldolgozóegysége, amely az összes feladatot (pl. műveletek elvégzése, változók összehasonlítása stb.) maga hajtja végre.

**Grafika** Számítógépes képek előállítása.

**Karakterlánc** Egy változóban tárolható karakterek sorozata, pl. "KARAKTERLÁNC" vagy "ABC123".

**Képpontok** (pixelek) Kis négyzetek, amelyeket a mikró egymástól függetlenül ki tud világítani — ezekből állítják össze a képeket.

**Kilobájt** (K) A számítógép memóriáját ebben a mértékegységben szokták mérni. 1024 bájtot (vagyis a legtöbb mikrónál 1024 karaktert) jelent.

**Kurzor** (őrszem) Egy kis villogó (legtöbbször négyzet alakú) jel, azt mutatja, hova íródik ki a következő karakter.

**LET** Megcímkéz egy memóriaterületet, és beleír valamilyen adatot. Pl. **LET N=4** vagy **LET B\$="MACSKA"**.

**\*LIST** Kilistázza a programot a képernyőre.

**\*MID\$** Egy karakterlánc megadott helyétől kezdve vesz néhány karaktert. Pl. **MID\$(A\$,4,3)** az **A\$ 4.** betűjével kezdődő három szomszédos karaktert jelenti.

**NEW** Kitörli a programot a gép memóriájából, hogy helyet biztosítson a következőnek.

**\*NEWLINE** gomb Egy begépelte programsor vagy adat végét jelzi. Egyes gépeken **RETURN** vagy **ENTER** felíratot visel.

**NEXT L. FOR.**

**\*PLOT** Egy képpontot kigyújt a képernyőn. **PLOT(X,Y)**-ban **X** a vízszintes, **Y** a függőleges koordinátát jelöli.

**PRINT** Valamit kiír a képernyőre.

**\*READ** Adatot olvas be egy változóba az éppen esedékes **DATA** sorból (l. **DATA**).

**\*READY** Egyes mikrók ezzel jelzik, hogy készen állnak egy újabb utasítás végrehajtására.

**REM** A gép figyelmen kívül hagyja a **REM**-mel kezdődő sorokat, csak a programlistán tünteti fel. A **REM** után tehát a program működésére emlékeztető szöveget írhatunk be.

**RETURN** Egy szubrutin végén visszaküldi a programot a szubrutint hívó **GOSUB**-ot követő utasításra (l. **GOSUB**).

**\*RIGHT\$** Egy karakterlánc végétől vesz megadott számú karaktert.

**\*RND** Egy véletlen számot állít elő.

**RUN** Elindítja a program végrehajtását.

**SQR** Egy szám négyzetgyökét veszi.

**STEP** A **FOR... ciklus** utasításban szerepelhet: az ismétlést számláló ciklusváltozó növekedését szabja meg.

**STOP** A program belsejében ezzel állíthatjuk meg a program futását.

**THEN L. IF.**

**\*UNPLOT** Kiolt egy kigyújtott képpontot.

**Program** Sorszámmal ellátott utasítások sorozata. Megmondja a számítógépnek, hogyan kell egy adott feladatot végrehajtania.

**RAM** Tetszőleges elérési memória. A számítógépnek az a memóriája, amely a programot és az adatokat tárolja. A gép kikapcsolásakor törlődik. Másképpen írható-olvasható memóriának mondható.

**ROM** Csak olvasható memória. A számítógép működését vezérlő állandó programot tárolja.

**Szintaktikai hiba** A BASIC (vagy más programnyelv) "nyelvtani" szabályait sértő programhiba.

**Szubrutin** Egy részfeladat megoldására szolgáló programrészlet, amely egy program futása során rendszerint többször is végrehajtható.

**Tömb** Azonok nevű változók csoportja, amelyek különböző adatokat tartalmazhatnak. A változónév után zárójelbe közt álló sorszámmal (indexszel) különböztetjük meg őket.

**Változó** Megcímkézett memóriaterület, ahol adatokat tárolhatunk.

# Hogyan tovább?

Programozni csak a gyakorlatban lehet megtanulni. Ha nincs saját géped, érdeklődj utána, hol tudnál egy kicsit géphez ülni! Az iskolában, művelődési házban, klubokban mindenkit szívesen fogadnak. Búcsúzóul még néhány könyvet ajánlunk.

Alcock: **Ismerd meg a BASIC nyelvet**

Kőhegyi (szerk.): **Ismerd meg a BASIC nyelvjárásait I (ZX 81, HT 1080Z, ABC 80)**

**Ismerd meg a BASIC nyelvjárásait II (ZX Spectrum, Proper, Texas)**

**Ismerd meg a BASIC nyelvjárásait III (C-64, VIC-20, SHARP 1500)**

Kocsis András: **TV BASIC**

**Hetedhét Commodore 64** Novotrade, 1985.

**Hetedhét Commodore 16** Novotrade, 1985.

**Hetedhét Atari** Novotrade, 1986.

**Csupa játék ZX Spectrumra** Műszaki Könyvkiadó, 1986.

*Sorozatunkban még megjelenik:*

**Első könyvem a mikrókról és Első könyvem a chipekről.**

## Tárgymutató

### ABS 31

angol lecke 18

animáció 36

arcprogram 11

BASIC 7, 9, 10, 36, 38, 42

beszélgetőprogram 21

beugratós program 26

billentyűzet 4

BREAK 15, 23, 35, 37, 47

bug 8

ciklusok 26 – 29, 33, 34, 35, 37, 39, 41, 43

ciklushibák 28

címkerés 12

CLS 10, 15, 20, 25, 27, 28, 29, 34, 37, 43, 47

COPY 11

DATA 13, 21, 31, 39, 40, 41, 43, 47

DELETE 11

DIM 40, 41, 42, 47

EDIT 11, 47

egymásba ágyazott ciklusok 28

END 11, 47

Életkorprogram 18

ENTER 10

értelmező 6, 10

ESCAPE 15, 25, 47

falánkok programja 27

finom felbontású grafikon 22, 29, 37

folyamatlábra 9

FOR...NEXT 26 – 29, 33, 34, 35, 37, 39, 41, 43, 47

GOSUB 30 – 31, 35, 37, 43, 47

GOTO 19, 20, 21, 23, 25, 31, 38, 40, 41, 43, 47

grafika 22 – 23, 25, 29, 31, 36 – 37, 43

grafikus tábla 23

hiba 8, 9, 11, 28, 40, 42, 43

hibaüzenet 11, 36, 43

IF...THEN 18 – 19, 20, 21, 23, 25,

28, 31, 35, 37, 38, 40, 41, 47

INPUT 10, 14, 15, 21, 47

INT 24, 47

karakter 12

karakteres változó 12

képpont 22

kód 6, 7, 36

korprogram 31

kurzor 10

LEFTS 32 – 33, 40, 47

LEN 32, 33, 47

LET 12, 13, 17, 32, 47

LIST 11, 15, 47

mono 23

memóriaterület 12

MIDS 32, 33, 47

mintaismétlő program 29

monitor 4

NEW 15, 47

NEWLINE 10, 15, 47

őrzőjelet az űrben 20

összeadás 16, 34

Pascal 7

pilot 7

PLOT 22 – 23, 25, 29, 34, 36, 37, 43, 47

PRINT 10 – 11, 12, 13, 16 – 17, 47

program 4, 6, 7, 8, 9

programelágazások 19

RAM 5

READ 13, 21, 31, 39, 40, 41, 43, 47, 47

READY 10

rejtvényprogram 31

REM 27, 30, 31, 35, 37, 43, 47

RETURN gomb 10, 47

RESET 22

RIGHTS 32, 33, 40, 47

RND 24 – 25, 28, 29, 37, 38, 40, 41, 43, 47

ROM 5

RUBOUT 11

RUN 10 – 11, 15, 47

Sebességátváltó program 31

SET 22

STEP 27, 29, 33, 47

STOP 19, 30, 31, 35, 38, 40, 41, 47

számítógépes óra 28

számláló 21, 23, 25, 26, 31, 35, 38, 40

számváltozó 12, 13, 14

szorzótábla 26

szövegváltozó 12

szubrutin 30, 31, 35, 37, 43

születésnapprogram 35

titkosításprogram 33

THEN 18

TO 32

tömb 39, 40, 41

UNPLOT 22, 36, 47

űrtámadás 25

változó 12 – 15, 17, 18, 21, 27, 32,

35, 38, 39, 40

véletlen mintarajzoló 25

véletlen szám 24 – 25, 28, 29, 37, 38, 40

véletlenszám-ellenőrző 28

versprogram 41

vessző 13, 16, 17, 28, 42, 43

vonalhúzó 37

Az eredeti könyv címe:

Introduction to computer programming

Írta: Brian Reffin Smith • Szerkesztette: Lisa Watts •

Tervezte: Kim Blundell • Illusztrálta: Graham Round és Martin Newton

© Usborne Publishing 1982 • All right reserved

Hungarian Translation

© Műszaki Könyvkiadó, Novotrade Rt., Budapest, 1987

Fordította: Bokor Judit • Szakmailag ellenőrizte: Kepes János

ISBN: 963 10 7167 7 • ETO: 809.92 BASIC



# SZÁMÍTÓGÉPES KÉPESKÖNYVEK

jelennek meg

a Műszaki Könyvkiadó és a Novotrade Rt.

közös gondozásában.

## ELSŐ KÖNYVEM

### A MIKRÓKRÓL

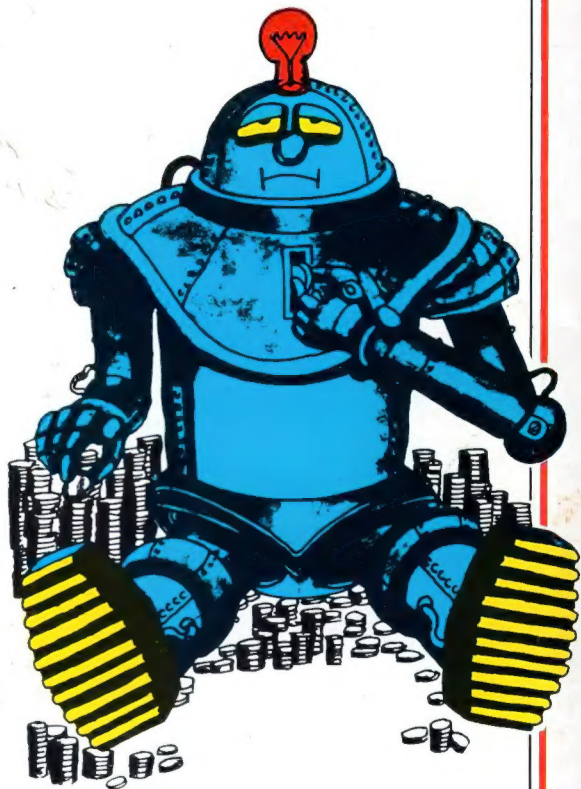
A mikroszámítógépek vagyis a mikrók ugyanúgy hozzá tartoznak a hétköznapiakhoz, mint az autó, a tv vagy a magnó. Mi mindenre képesek? Hogyan működnek? Miből épülnek föl? Mi volt a történeti fejlődésük és még mi várható?

– kérdésekre ad választ ez a kötet

## ELSŐ KÖNYVEM

### A PROGRAMOZÁSRÓL

A mikrókat programokkal irányíthatjuk. Már az elemi BASIC-nyelv segítségével is sok érdekességre képesek. Rajzolnak és verset írnak, látszólag értelmesen beszélgetnek, szerencsejátékokat és üldözéssel játszókat utánoznak.



Az eredeti könyvek  
az USBORNE gyermekkönyvkiadónál jelentek meg.